```
7bf681 1 MANIFEST
d9b4d3 D books/
ceb2fd D books/tools/
b88b8c 2 bootstrap.pl
9cc632 3 bootstrap2.pl
e2f823 4 sortpages.pl
9c2275 5 Makefile
b6fØe8 6 heap.c
b859bd 7 heap.h
dØ496Ø 8 mempool.c
48289d 9 mempool.h
a1d59f 1Ø util.c
6d7cca 11 util.h
3bec44 12 repair.c
fØ8499 13 subst.c
81a4c4 14 subst.h
176cØf 15 unmunge.c
ØeØ4e9 16 munge.c
4f3b67 17 yapp.doc
c52b17 18 yapp.pl
1873dd 19 psgen.pl
29287d 2Ø makemanifest.pl
97db93 D books/ps/
Ø26ff1 21 prolog.ps
2413dØ 22 charmap.ps
29c361 D books/example/
22ebd1 23 Makefile
3d6773 24 .gitignore
95a545 25 filelist
5ef59Ø 26 footer.ps
e3ad66 27 us-constitution.gz
```

Directory

books/

Directory

books/tools/

```
a273b6 #!/usr/bin/env perl -s
cfa6Ø1 #
8d4467 # bootstrap -- Simpler version of unmunge for bootstrapping
67a6Ø1 #
4a33f9 # Unmunge this file using:
9837c3 # ‘‘perl -ne 'if (s/^ *[^-\s]\S{4,6} ?//) { s/[\244\245\267]/ /g; print; }'
2ca6Ø1 #
8Ø8bb8 # $Id: bootstrap,v 1.15 1997/11/14 Ø3:52:53 mhw Exp $
c6af5a
9c1496 sub Fatal▷      { print STDERR @␣; ‘exit(1); }
1e0a97 sub Max▷▷       { my ($a, $b) = @␣; ‘($a > $b) ? $a : $b; }
fa36b1 sub TabSkip▷    { $tabWidth - 1 - (length($␣[Ø]) % $tabWidth); }
7faf5a
534cdd ($tab,$yen,$pilc,$cdot,$tmp1,$tmp2)=("\244","\245","\266","\267","\377","\376");
Øb3Ø67 $editor = $ENV{'VISUAL'} ¦¦ $ENV{'EDITOR'} ¦¦ 'vi';
cde7e6 $inFile = $ARGV[Ø];
cc4e2f doFile: {
78e81b ‘‘‘‘open(IN, "<$inFile") ¦¦ die;
e45163 ‘‘‘‘for ($lineNum = 1; ($␣ = <IN>); $lineNum++) {
41aacd ▷       s/^\s+//; ‘s/\s+$//;▷   # Strip leading and trailing spaces
4ab32f ▷       next if (/^$/);▷▷       # Ignore blank lines
41f118 ▷       ($prefix, $seenCRCStr, $dummy, $␣) = /^(\S{2})(\S{4})( (.*))?/;
bcaf5a
e5b8ac ▷       # Correct the number of spaces after each tab
ea2d31 ▷       while (s/$tab( *)/$tmp1 . ($tmp2 x &Max(length($1), &TabSkip($`)))/e) {}
763e7b ▷       s/ ( +)/" " . ($cdot x length($1))/eg;▷ # Correct center dots
dØØ24c ▷       s/$tmp1/$tab/g; ‘s/$tmp2/ /g; ‘# Restore tabs and spaces from correction
543cdØ ▷       s/\s*$/\n/;▷    ▷       # Strip trailing spaces, and add a newline
Øbaf5a
d72516 ▷       $crc = $seenCRC = Ø;▷   ▷       ▷       # Calculate CRC
293db3 ▷       for ($data = $␣; $data ne ""; $data = substr($data, 1)) {
a4eØØ2 ▷       ‘‘‘‘$crc ^= ord($data);
6Ø2ea8 ▷       ‘‘‘‘for (1..8) {
78Øb86 ▷       ▷       $crc = ($crc >> 1) ^ (($crc & 1) ? Øx84Ø8 : Ø);
2c4d5a ▷       ‘‘‘‘}
191aea ▷       }
234656 ▷       if ($crc != hex($seenCRCStr)) {▷▷       # CRC mismatch
d25ba7 ▷       ‘‘‘‘close(IN); ‘close(OUT);
2c4921 ▷       ‘‘‘‘unlink(@filesCreated);
b7fc5c ▷       ‘‘‘‘@filesCreated = ();
28ce26 ▷       ‘‘‘‘@oldStat = stat($inFile);
41e5bØ ▷       ‘‘‘‘system($editor, "+$lineNum", $inFile);
d11549 ▷       ‘‘‘‘@newStat = stat($inFile);
c593df ▷       ‘‘‘‘redo doFile if ($oldStat[9] != $newStat[9]); ‘# Check mod date
b9aØde ▷       ‘‘‘‘&Fatal("Line $lineNum invalid: $␣");
6e1aea ▷       }
31af5a
387b7d ▷       if ($prefix eq '--') {▷ ▷       ▷       # Process header line
3Ø541c ▷       ‘‘‘‘($code, $pageNum, $file) = /^(\S{19}) Page (\d+) of (.*)/;
1d88c3 ▷       ‘‘‘‘$tabWidth = hex(substr($code, 11, 1));
627dØ1 ▷       ‘‘‘‘if ($file ne $lastFile) {
5f1aaf ▷       ▷       print "$file\n";
bb8e77 ▷       ▷       &Fatal("$file: already exists\n") if (!$f && (-e $file));
4ba6b3 ▷       ▷       close(OUT);
b6Ø97b ▷       ▷       open(OUT, ">$file") ¦¦ &Fatal("$file: $!\n");
5dbd21 ▷       ▷       push(@filesCreated, ($lastFile = $file));
3f4d5a ▷       ‘‘‘‘}
e74ccd ▷       } else {▷       ▷       ▷       ▷       # Unmunge normal line
685291 ▷       ‘‘‘‘s/$tab( *)/"\t".(" " x (length($1) - &TabSkip($`)))/eg;
6eb547 ▷       ‘‘‘‘s/$yen\n/\f/;▷      # Handle form feeds
df7ad4 ▷       ‘‘‘‘s/$pilc\n//;▷       # Handle continuation lines
ac9f82 ▷       ‘‘‘‘s/$cdot/ /g;▷       # Center dots -> spaces
8daf5a
cfe546 ▷       ‘‘‘‘print OUT;
3d1aea ▷       }
6c6fe7 ‘‘‘‘}
4acaØ6 ‘‘‘‘close(IN); ‘close(OUT);
5aefe6 }
```

```
a273b6 #!/usr/bin/env perl -s
cfa6Ø1 #
264352 # bootstrap2 -- Second stage bootstrapper, a version of unmunge
31a6Ø1 #
739cbb # $Id: bootstrap2,v 1.4 1997/11/14 Ø3:52:54 mhw Exp $
Øbaf5a
56d22f sub Cleanup▷    { close(IN); ‧close(OUT); ‧unlink(@files); ‧@files = (); }
f42a1e sub Fatal▷      { &Cleanup(); ‧print STDERR @_; ‧exit(1); }
4236b1 sub TabSkip▷    { $tabWidth - 1 - (length($_[Ø]) % $tabWidth); }
a2172b sub TabFix▷     { my ($needed, $actual) = (&TabSkip($_[Ø]), length($_[1]));
ØØ5323 ‧‧‧‧$tmp1 . ($tmp2 x $needed) . (" " x ($actual - $needed)); }
7a2Øf4 sub HumanEdit▷  { my ($file, $line, @message) = ($inFile, @_); ‧&Cleanup();
a82db1 ‧‧‧‧@old = stat($file); ‧system($editor, "+$line", $file); ‧@new = stat($file);
c277e8 ‧‧‧‧redo doFile if ($old[9] != $new[9]);▷       # Check mod date
517c59 ‧‧‧‧&Fatal("Line $line, ", @message); }
9caf5a
144cdd ($tab,$yen,$pilc,$cdot,$tmp1,$tmp2)=("\244","\245","\266","\267","\377","\376");
d13Ø67 $editor = $ENV{'VISUAL'} ¦¦ $ENV{'EDITOR'} ¦¦ 'vi';
9da6f7 ($inFile, $manifest, @rest) = @ARGV;
c3bb7Ø if ($manifest ne "") {▷ ▷       # Read manifest file
ab997Ø ‧‧‧‧open(MANIFEST, "<$manifest") ¦¦ &Fatal("$manifest: $!\n");
aØe3e3 ‧‧‧‧while (<MANIFEST>) { $dir = $1 if /^D\s+(.*)$/;
7e9857 ▷       $index[$1] = $dir . $2 if /^(\d+)\s+(.*)$/; }
92efe6 }
9a4e2f doFile: {
Ød779a ‧‧‧‧$seenPCRC = $pcrc1 = Ø; ‧$lastFlags = 1; ‧$lastFileNum = Ø;
d42616 ‧‧‧‧open(IN, "<$inFile") ¦¦ &Fatal("$inFile: $!\n");
ØØc787 ‧‧‧‧for ($line = 1; ($_ = <IN>); $line++) {
18aacd ▷       s/^\s+//; ‧s/\s+$//;▷   # Strip leading and trailing spaces
dab32f ▷       next if (/^$/);▷▷       # Ignore blank lines
1df118 ▷       ($prefix, $seenCRCStr, $dummy, $_) = /^(\S{2})(\S{4})( (.*))?/;
633e5a ▷       while (s/$tab( *)/&TabFix($`, $1)/eo) {} ‧# Correct spaces after tabs
Øbdb12 ▷       s/($tmp2¦ )( +)/$1 . ($cdot x length($2))/ego;▷ # Correct center dots
Ø94668 ▷       s/$tmp1/$tab/go; ‧s/$tmp2/ /go; ‧# Restore tabs/spaces from correction
313cdØ ▷       s/\s*$/\n/;▷    ▷       # Strip trailing spaces, and add a newline
dfaf5a
b6Ø46Ø ▷       $crc = Ø; ‧$pcrc = $pcrc1;▷     ▷       # Calculate CRCs
fa3db3 ▷       for ($data = $_; $data ne ""; $data = substr($data, 1)) {
ba6Øae ▷       ‧‧‧‧$crc ^= ord($data); ‧$pcrc1 ^= ord($data);
9c28fØ ▷       ‧‧‧‧for (1..8) { $crc = ($crc >> 1) ^ (($crc & 1) ? Øx84Ø8 : Ø);
4fØØf2 ▷       ▷       $pcrc1 = ($pcrc1 >> 1) ^ (($pcrc1 & 1) ? Øxedb8832Ø : Ø); }
551aea ▷       }
38e7eb ▷       ($seenPLCRC, $seenCRC) = map { hex($_) } ($prefix, $seenCRCStr);
e24eda ▷       &HumanEdit($line, "CRC failed: $_") if $crc != $seenCRC;
9Ø7b7d ▷       if ($prefix eq '--') {▷ ▷       ▷       # Process header line
ba2129 ▷       ‧‧‧‧&HumanEdit($line - 1, "Page CRC failed") if $pcrc != $seenPCRC;
Ø9991f ▷       ‧‧‧‧($humanHdr, $pageNum, $file) = /^\S{19} (Page (\d+) of (.*))/;
d9371Ø ▷       ‧‧‧‧($vers, $flags, $seenPCRC, $tabWidth, $prodNum, $fileNum) =
a82c3f ▷       ▷       map { hex($_) } /^(\S)(\S\S)(\S{8})(\S)(\S{3})(\S{4})/;
97Øb72 ▷       ‧‧‧‧if ($fileNum != $lastFileNum) {
117Øbd ▷       ▷       print STDERR "MISSING files\n" if $fileNum != $lastFileNum + 1;
b861Ø2 ▷       ▷       &Fatal("Missing pages\n") if $pageNum != 1 ¦¦ !($lastFlags & 1);
526aeb ▷       ▷       if ($manifest ne "") {
ddfd6f ▷       ▷       ‧‧‧‧($_ = $index[$fileNum]) =~ m%([^/]*)$%;
4dae35 ▷       ▷       ‧‧‧‧&Fatal("Manifest mismatch\n") if ($file ne $1);
e85Ød2 ▷       ▷       ‧‧‧‧($file = $_) =~ s¦/+¦mkdir($`, Ø777), "/"¦eg; ‧# mkdir -p
cd467a ▷       ▷       }
5f8e77 ▷       ▷       &Fatal("$file: already exists\n") if (!$f && (-e $file));
ed5c6f ▷       ▷       close(OUT); ‧open(OUT, ">$file") ¦¦ &Fatal("$file: $!\n");
73bØ66 ▷       ▷       push(@files, $file); ‧print "$fileNum $file\n";
9b9957 ▷       ‧‧‧‧} else {
56efb5 ▷       ▷       &Fatal("MISSING pages\n") if ($pageNum != $lastPageNum + 1);
a44d5a ▷       ‧‧‧‧}
85ba7e ▷       ‧‧‧‧($lastFlags,$lastFileNum,$lastPageNum) = ($flags,$fileNum,$pageNum);
1fe8Ø9 ▷       ‧‧‧‧$pcrc1 = Ø;
f44ccd ▷       } else {▷       ▷       ▷       ▷       # Unmunge normal line
721c35 ▷       ‧‧‧‧&HumanEdit($line, "CRC failed: $_") if ($pcrc1 >> 24) != $seenPLCRC;
Øc65fØ ▷       ‧‧‧‧s/$tab( *)/"\t".(" " x (length($1) - &TabSkip($`)))/ego;
25c825 ▷       ‧‧‧‧s/$yen\n/\f/o; ‧s/$pilc\n//o; ‧s/$cdot/ /go; ‧print OUT;
451aea ▷       }
766fe7 ‧‧‧‧}
efefe6 }
```

```
Øb326b #!/usr/bin/env perl
44a6Ø1 #
5a82fe # $Id: sortpages,v 1.8 1997/12/11 19:2Ø:58 mhw Exp $
44a6Ø1 #
b3af5a
5Øbbf4 @fileNameFromNumber = ();
6842b8 @pagesFound = ();
4ØØcca $theProductNumber = Ø;
85af5a
735ØØf for $fileIndex (Ø..$#ARGV)
56bb36 {
7d3251 ▷   $fileName = $ARGV[$fileIndex];
3d543Ø ▷   open(FILE, "<$fileName") ¦¦ die;
45b525 ▷   while (!eof(FILE))
21d827 ▷▷▷ {
bf6e8Ø ▷▷▷ ▷   $filePos = tell(FILE);
a5Ø63b ▷▷▷ ▷   $_ = <FILE>;
e87ef3 ▷▷  ▷   if (/^\f?-\S/)
9ff292 ▷▷▷ ▷   {
f8d62e ▷▷▷ ▷   ▷   my ($versionHex, $flagsHex, $pageCRCHex, $tabWidthHex,
6d79Ø5 ▷   ▷   ▷   ▷   $productNumberHex, $fileNumberHex, $pageNumber, $name)
b72de7 ▷   ▷   ▷   ▷   ▷   ▷▷= (/^\f?-\S\S{4}\ ▷   ▷   # CRC followed by a space
dcØdc7 ▷   ▷   ▷   ▷   ▷   ▷   ▷([Ø-9a-f])▷▷   ▷   ▷   # Format version
75d528 ▷   ▷   ▷   ▷   ▷   ▷   ▷([Ø-9a-f]{2})▷ ▷   ▷   # Flags
215e3d ▷   ▷   ▷   ▷   ▷   ▷   ▷([Ø-9a-f]{8})▷ ▷   ▷   # Running CRC32
2Ø2d5c ▷   ▷   ▷   ▷   ▷   ▷   ▷([Ø-9a-f])▷▷   ▷   ▷   # Tab width (Ø means radix64)
6ea477 ▷   ▷   ▷   ▷   ▷   ▷   ▷([Ø-9a-f]{3})▷ ▷   ▷   # Product number
887ef1 ▷   ▷   ▷   ▷   ▷   ▷   ▷([Ø-9a-f]{4})▷ ▷   ▷   # File number
fØ59c2 ▷   ▷   ▷   ▷   ▷   ▷   ▷\ Page\ (\d+)\ of\ (.*)/x);
8e5c49 ▷   ▷   ▷   my $version = hex($versionHex);
224e6d ▷   ▷   ▷   my $flags = hex($flagsHex);
898e5d ▷   ▷   ▷   my $productNumber = hex($productNumberHex);
abe5a9 ▷   ▷   ▷   my $fileNumber = hex($fileNumberHex);
deaf5a
e4e898 ▷   ▷   ▷   unless ($version == Ø && $productNumber > Ø
72Ø639 ▷   ▷   ▷   ▷   ▷   ▷   && $fileNumber > Ø && $pageNumber > Ø
3d2ØØd ▷   ▷   ▷   ▷   ▷   ▷   && $name ne "")
3ef776 ▷   ▷   ▷   {
5bØb1e ▷   ▷   ▷   ▷   print STDERR "ERROR: Invalid header info ",
f92ce4 ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷"at $fileName line $.\n";
41bbe9 ▷   ▷   ▷   ▷   exit(1);
f5a3a6 ▷   ▷   ▷   }
cfaf5a
831aØ1 ▷   ▷   ▷   if (!defined($fileNameFromNumber[$fileNumber]))
6bf776 ▷   ▷   ▷   {
5cf9d6 ▷   ▷   ▷   ▷   $fileNameFromNumber[$fileNumber] = $name;
72a3a6 ▷   ▷   ▷   }
4Ø731b ▷   ▷   ▷   elsif ($fileNameFromNumber[$fileNumber] ne $name)
dff776 ▷   ▷   ▷   {
2Ø1def ▷   ▷   ▷   ▷   print STDERR "ERROR: Mismatched filename ",
572ce4 ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷"at $fileName line $.\n";
fabbe9 ▷   ▷   ▷   ▷   exit(1);
85a3a6 ▷   ▷   ▷   }
Ø6af5a
cbe89Ø ▷   ▷   ▷   if (!$theProductNumber)
16f776 ▷   ▷   ▷   {
Ø71272 ▷   ▷   ▷   ▷   $theProductNumber = $productNumber;
cba3a6 ▷   ▷   ▷   }
d2aaØ5 ▷   ▷   ▷   elsif ($theProductNumber != $productNumber)
2ef776 ▷   ▷   ▷   {
935f14 ▷   ▷   ▷   ▷   print STDERR "ERROR: Different product number ",
c82ce4 ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷"at $fileName line $.\n";
fabbe9 ▷   ▷   ▷   ▷   exit(1);
27a3a6 ▷   ▷   ▷   }
3faf5a
4bc93a ▷   ▷   ▷   push @pagesFound, (sprintf "%5d:%4d:%d:%d:%d",
bdf69Ø ▷   ▷   ▷   ▷   ▷   ▷$fileNumber, $pageNumber, $flags, $fileIndex, $filePos);
c7Øfe4 ▷   ▷   }
449371 ▷   }
93d33d ▷   close(FILE) ¦¦ die;
dfefe6 }
55af5a
```

```
d819ef @pagesFound = sort @pagesFound;
ceaf5a
3e64fe $result = Ø;
5fafd8 $lastFileNumber = Ø;
39Øf2c $lastPageNumber = Ø;
87eØc8 $nextFileNumber = 1;
534Ø3c $nextPageNumber = 1;
4d8Øf5 $fileIndexOpen = -1;
d4ea12 foreach (@pagesFound)
63bb36 {
7d7635 ▷   my ($fileNumber, $pageNumber, $flags, $fileIndex, $filePos) = split /:/;
c6af5a
b6a54b ▷   $fileNumber = int($fileNumber);
73eaa2 ▷   $pageNumber = int($pageNumber);
beaf5a
74fec7 ▷   if ($fileNumber == $lastFileNumber && $pageNumber == $lastPageNumber)
eec7a1 ▷   {
6Ø826Ø ▷   ▷   print STDERR "DUPLICATE: File $fileNumber, page $pageNumber, skipped\n";
a2a857 ▷   ▷   next;
c39371 ▷   }
43af5a
74Ø17f ▷   if ($nextFileNumber < $fileNumber && $nextPageNumber != 1)
Ø9c7a1 ▷   {
7cced2 ▷   ▷   print STDERR "MISSING: File $nextFileNumber, ",
Ø2431Ø ▷   ▷   ▷   ▷   ▷   ‘"pages $nextPageNumber - END\n";
f4c5bØ ▷   ▷   $nextPageNumber = 1;
Ø4ee6a ▷   ▷   $nextFileNumber++;
16ca38 ▷   ▷   $result = 1;
Øf9371 ▷   }
75c6b8 ▷   if ($nextFileNumber < $fileNumber)
Ø3c7a1 ▷   {
e7f2Ø8 ▷   ▷   print STDERR "MISSING: Files $nextFileNumber - ",
f2e199 ▷   ▷   ▷   ▷   ▷   ‘$fileNumber-1, "\n";
481bdb ▷   ▷   $nextFileNumber = $fileNumber;
86c5bØ ▷   ▷   $nextPageNumber = 1;
c1ca38 ▷   ▷   $result = 1;
269371 ▷   }
7c4a96 ▷   if ($nextFileNumber != $fileNumber)
8cc7a1 ▷   {
38Øb33 ▷   ▷   print STDERR "ERROR: Internal error, unexpected fileNumber\n";
Øc4bb3 ▷   ▷   exit(1);
9f9371 ▷   }
eeaf5a
917b5f ▷   if ($nextPageNumber < $pageNumber)
6fc7a1 ▷   {
a88Øef ▷   ▷   print STDERR "MISSING: File $fileNumber, pages $nextPageNumber - ",
7beb89 ▷   ▷   ▷   ▷   ▷   ‘$pageNumber-1, "\n";
49a63c ▷   ▷   $nextPageNumber = $pageNumber;
72ca38 ▷   ▷   $result = 1;
fb9371 ▷   }
3165ae ▷   if ($nextPageNumber != $pageNumber)
54c7a1 ▷   {
Ødddc6 ▷   ▷   print STDERR "ERROR: Internal error, unexpected pageNumber\n";
324bb3 ▷   ▷   exit(1);
Øa9371 ▷   }
67af5a
21bdØ3 ▷   if ($fileIndexOpen != $fileIndex)
66c7a1 ▷   {
23d16a ▷   ▷   if ($fileIndexOpen >= Ø)
725b34 ▷   ▷   {
ed88Ø6 ▷   ▷   ▷   close(FILE) ¦¦ die;
79be2f ▷   ▷   ▷   $fileIndexOpen = -1;
f9Øfe4 ▷   ▷   }
3373a2 ▷   ▷   $fileName = $ARGV[$fileIndex];
cØ51dd ▷   ▷   open(FILE, "<$fileName") ¦¦ die;
Ø395f2 ▷   ▷   $fileIndexOpen = $fileIndex;
cØ9371 ▷   }
936cec ▷   seek(FILE, $filePos, Ø) ¦¦ die($!);
ddaf5a
Øa891b ▷   $_ = <FILE>;
43c63b ▷   print;
b58f9e ▷   while (<FILE>)
```

```
f7c7a1 ▷   {
Ø737ea ▷   ▷   last if /^\f?-\S/;
c7a781 ▷   ▷   print;
969371 ▷   }
4ffd9c ▷   $lastFileNumber = $fileNumber;
454Ø7b ▷   $lastPageNumber = $pageNumber;
76af5a
25bc6e ▷   if ($flags & 1)▷▷   # Bit Ø of flags indicates last page of file
4Øc7a1 ▷   {
f6ee6a ▷   ▷   $nextFileNumber++;
9bc5bØ ▷   ▷   $nextPageNumber = 1;
e39371 ▷   }
3cØe24 ▷   else
2ec7a1 ▷   {
3f96ba ▷   ▷   $nextPageNumber++;
ee9371 ▷   }
48efe6 }
bdaf5a
ee172e if ($nextPageNumber != 1)
3Øbb36 {
3Ø9b7e ▷   print STDERR "MISSING: File $nextFileNumber, ",
cced11 ▷   ▷   ▷   ▷   ‘"pages $nextPageNumber - END\n";
48f1ed ▷   $nextPageNumber = 1;
51Øc8b ▷   $nextFileNumber++;
968bb3 ▷   $result = 1;
b9efe6 }
16af5a
cb817c print STDERR "Highest file number encountered: ", $nextFileNumber - 1, "\n";
e6af5a
7a5e61 if ($fileIndexOpen >= Ø)
c9bb36 {
c3d33d ▷   close(FILE) ¦¦ die;
5c3124 ▷   $fileIndexOpen = -1;
d1efe6 }
c3af5a
9f7c7Ø exit($result);
baaf5a
8ba6Ø1 #
93f1a9 # vi: ai ts=4
c8887e # vim: si
e1a6Ø1 #
```

```
--4f77 ØØ171eda43f8ØØ2ØØØ5 Page 1 of Makefile

52951b BIN ⋆= bin
dØc599 OBJ ⋆= obj
6cdce2 DIRS = $(BIN) $(OBJ)
c8a897 CFLAGS = -g -O -W -Wall
96af5a
177f86 PERL ⋆⋆⋆⋆⋆⋆⋆⋆= $(addprefix $(BIN)/, bootstrap bootstrap2 makemanifest psgen sor
b279b9 tpages yapp)
c5f2f3 BINS ⋆⋆⋆⋆⋆⋆⋆⋆= $(addprefix $(BIN)/, unmunge munge repair)
Øe82bb UNMUNGE_OBJS = $(addprefix $(OBJ)/, util.o unmunge.o)
eb9df2 MUNGE_OBJS ⋆⋆= $(addprefix $(OBJ)/, util.o munge.o)
44c566 REPAIR_OBJS ⋆= $(addprefix $(OBJ)/, util.o heap.o mempool.o subst.o repair.o)
4eaf5a
81275b $(shell mkdir -p $(DIRS))
35af5a
bd62ed all: $(BINS) $(PERL)
e9af5a
5d8c53 $(BIN)/%: %.pl
98fbf4 ▷       cp $< $@; chmod +x $@
ecaf5a
adfb7a $(BIN)/unmunge: $(UNMUNGE_OBJS)
f599eb ▷       $(CC) $(CFLAGS) -o $@ $(UNMUNGE_OBJS)
b3af5a
a47aea $(BIN)/munge: $(MUNGE_OBJS)
1d37f8 ▷       $(CC) $(CFLAGS) -o $@ $(MUNGE_OBJS)
Ødaf5a
55693d $(BIN)/repair: $(REPAIR_OBJS)
b39e3c ▷       $(CC) $(CFLAGS) -o $@ $(REPAIR_OBJS)
23af5a
9cbcØ6 $(OBJ)/%.o: %.c
29f338 ▷       $(CC) $(CFLAGS) -c -o $@ $<
f4af5a
f449c8 clean:
6Ø9eff ▷       rm -f $(OBJ)/* $(BIN)/*.core
7aaf5a
Øe1ed2 cleaner:
715bf2 ▷       rm -rf $(DIRS)
```

```
bc38e5 /*
5d7856 ·* heap.c -- Simple priority queue. ·Takes pointers to cost values
7cØb95 ·* (presumably the first field in a larger structure) and returns
4d3169 ·* them in increasing order of cost.
d3775e ·*
52a5ec ·* Copyright (C) 1997 Pretty Good Privacy, Inc.
18775e ·*
3e3157 ·* Written by Colin Plumb and Mark H. Weaver
6c775e ·*
4Øae23 ·* $Id: heap.c,v 1.2 1997/Ø7/Ø5 Ø2:55:23 colin Exp $
61495d ·*/
41af5a
6Ø5925 #include <stdio.h>▷ /* For fprintf(stderr, "Out of memory") */
1eddae #include <stdlib.h>▷/* For malloc() & co. */
Ø3af5a
b76Ø9b #include "heap.h"
d9af5a
a39ee3 #define HeapParent(i)▷  ▷   ▷   ((i) / 2)
de29cc #define HeapLeftChild(i)▷   ▷   ((i) * 2)
cbbØ41 #define HeapRightChild(i)▷  ▷   ((i) * 2 + 1)
a24665 #define HeapElem(h, i)▷ ▷   ▷   (h)->elems[i]
4d7ded #define HeapMinElem(h)▷ ▷   ▷   HeapElem(h, 1)
6dab41 #define HeapElemCost(e)▷▷   ▷   (*(e))
c1a86a #define HeapCost(h, i)▷ ▷   ▷   HeapElemCost(HeapElem(h, i))
77e7a5 #define HeapSize(h)▷▷   ▷   ▷   ((h)->numElems)
33af5a
ef2978 static void
abb5fe SiftDown(Heap const *heap, HeapCost *e)
39bb36 {
3ccead ▷   HeapIndex size = HeapSize(heap), parent = 1, child;
79f269 ▷   HeapCost cparent = HeapElemCost(e), cchild;
76af5a
9e22Øe ▷   for (;;) {
fad6cc ▷   ▷   child = 2*parent;
34a48b ▷   ▷   if (child > size)
62cdcb ▷   ▷   ▷   break;
911741 ▷   ▷   cchild = HeapCost(heap, child);
7d9561 ▷   ▷   if (child < size && cchild > HeapCost(heap, child+1)) {
fcd96c ▷   ▷   ▷   cchild = HeapCost(heap, child+1);
3Ø8a12 ▷   ▷   ▷   child++;
6ØØfe4 ▷   ▷   }
df9681 ▷   ▷   if (cparent <= cchild)
14d8a9 ▷   ▷   ▷   break;▷ /* Stop sifting down */
999Ø19 ▷   ▷   HeapElem(heap, parent) = HeapElem(heap, child);
b4a65b ▷   ▷   parent = child;
ff9371 ▷   }
Ø15Ød3 ▷   HeapElem(heap, parent) = e;
38efe6 }
33af5a
262fac /* Debug tool: verify heap property */
ae314a void
489c5e HeapVerify(Heap *heap)
Ø1bb36 {
ce99c7 ▷   HeapIndex i;
feaf5a
56ef31 ▷   for (i = 2; i <= HeapSize(heap); i++)
594317 ▷   ▷   if (HeapCost(heap, i) < HeapCost(heap, HeapParent(i)))
43f115 ▷   ▷   ▷   fprintf(stderr, "DEBUG: VerifyHeap failed at elem %d\n", i);
f1efe6 }
13af5a
f5e54Ø /* Remove and return the minimum cost from the heap. */
1943eØ HeapCost *
8977ba HeapGetMin(Heap *heap)
8dbb36 {
c1e922 ▷   HeapIndex lastElem = HeapSize(heap);
ecc46a ▷   HeapCost *retval;
c4af5a
b174aØ ▷   if (!lastElem)
Ø37ed2 ▷   ▷   return NULL;
5331af ▷   retval = HeapMinElem(heap);
76b354 ▷   HeapSize(heap) = lastElem-1;
82Ø26e ▷   SiftDown(heap, HeapElem(heap, lastElem));
```

```
7594b6 ▷   return retval;
d6efe6 }
59af5a
7b36bd /* Helper - set heap size, reallocating if needed */
bb2978 static void
5c9256 HeapResize(Heap *heap, HeapIndex newNumElems)
babb36 {
80d0b1 ▷   if (newNumElems >= heap->elemsAllocated) {
58dbc6 ▷   ▷   HeapIndex newAllocSize = heap->elemsAllocated * 2;
9caf5a
09a0a8 ▷   ▷   if (newAllocSize <= newNumElems)
64a954 ▷   ▷   ▷   newAllocSize = newNumElems + 1;
7a69de ▷   ▷   heap->elems = (HeapCost **)realloc((void *)heap->elems,
a34b02 ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ··sizeof(*heap->elems) * newAllocSize);
5342a8 ▷   ▷   if (heap->elems == NULL) {
9c4110 ▷   ▷   ▷   fprintf(stderr, "Fatal error: Out of memory growing heap\n");
a10a38 ▷   ▷   ▷   exit(1);
080fe4 ▷   ▷   }
73cf66 ▷   ▷   heap->elemsAllocated = newAllocSize;
d59371 ▷   }
df0328 ▷   heap->numElems = newNumElems;
71efe6 }
13af5a
61baee /* Add an element to the heap */
95314a void
0ae842 HeapInsert(Heap *heap, HeapCost *newElem)
f7bb36 {
0b991d ▷   HeapIndex parent, i = ++HeapSize(heap);
416276 ▷   HeapCost cost = HeapElemCost(newElem);
22af5a
41ddde ▷   HeapResize(heap, i);
66dfee ▷   /* Sift up until parent = 0 */
3cff20 ▷   while ((parent = HeapParent(i)) && HeapCost(heap, parent) > cost) {
dc5d88 ▷   ▷   HeapElem(heap, i) = HeapElem(heap, parent);
d730ba ▷   ▷   i = parent;
279371 ▷   }
309899 ▷   heap->elems[i] = newElem;
23efe6 }
2baf5a
0f7777 /* Initialize a new heap */
f0314a void
df81c9 HeapInit(Heap *heap, HeapIndex initSize)
bebb36 {
b585c5 ▷   initSize++;▷/* Add one for temporary element */
4d7d54 ▷   if (initSize < 1)
2570e2 ▷   ▷   initSize = 1;
d4ff50 ▷   heap->elems = (HeapCost **)malloc(initSize * sizeof(*heap->elems));
5fced7 ▷   if (heap->elems == NULL) {
cd84e5 ▷   ▷   fprintf(stderr, "Fatal error: Out of memory creating heap\n");
0f4bb3 ▷   ▷   exit(1);
949371 ▷   }
8b591e ▷   heap->elemsAllocated = initSize;
129335 ▷   heap->numElems = 0;
7befe6 }
53af5a
e38353 /* Free up a heap's resources. */
df314a void
abae85 HeapDestroy(Heap *heap)
d5bb36 {
f7f8c3 ▷   free((void *)heap->elems);
14246f ▷   heap->elemsAllocated = 0;
a89335 ▷   heap->numElems = 0;
7cb847 ▷   heap->elems = NULL;
52efe6 }
d8af5a
2138e5 /*
14e6c5 ·* Local Variables:
fc9b19 ·* tab-width: 4
43e7a4 ·* End:
5eb612 ·* vi: ts=4 sw=4
e96c42 ·* vim: si
be495d ·*/
```

```
bc38e5 /*
1a1395 ‣* heap.h -- Simple priority queue. ‣Takes pointers to cost values
fcØb95 ‣* (presumably the first field in a larger structure) and returns
b13169 ‣* them in increasing order of cost.
68775e ‣*
57a5ec ‣* Copyright (C) 1997 Pretty Good Privacy, Inc.
59775e ‣*
573157 ‣* Written by Colin Plumb and Mark H. Weaver
b4775e ‣*
1Ø6dØ4 ‣* $Id: heap.h,v 1.6 1997/1Ø/31 Ø4:22:46 mhw Exp $
9a495d ‣*/
43af5a
34b51e #ifndef HEAP␣H
7411ad #define HEAP␣H 1
4daf5a
71feb2 #include <stdio.h>
51bea3 #include <stdlib.h>
ØØa7aØ #include <limits.h>
f4af5a
efcc46 typedef int HeapCost;
58eee4 #define COST␣INFINITY INT␣MAX
d21fbd typedef unsigned HeapIndex;
aØaf5a
1754f2 typedef struct Heap {
bc3952 ▷   HeapCost▷   **elems;
b48725 ▷   HeapIndex▷  numElems, elemsAllocated;
1ØØd9e } Heap;
eØaf5a
e59b92 void HeapInit(Heap *heap, HeapIndex initSize);
43b26Ø void HeapDestroy(Heap *heap);
2e7fef void HeapInsert(Heap *heap, HeapCost *newElem);
cd9Øf5 HeapCost *HeapGetMin(Heap *heap);
e8e6Ø5 void HeapVerify(Heap *heap);
6daf5a
237454 #endif
b2af5a
9738e5 /*
15e6c5 ‣* Local Variables:
cb9b19 ‣* tab-width: 4
65e7a4 ‣* End:
c9b612 ‣* vi: ts=4 sw=4
ad6c42 ‣* vim: si
35495d ‣*/
```

```
bc38e5 /*
972ØØ5 ·* mempool.c - Pooled memory allocation, similar to GNU obstacks.
26775e ·*
1Ø3Ø4a ·* $Id: mempool.c,v 1.5 1997/11/13 23:53:Ø8 colin Exp $
76495d ·*/
b4bb5f #include <assert.h>
Ø9feb2 #include <stdio.h>
ca324c #include <string.h>
Øe9126 #include <stdlib.h>▷    /* For malloc() & free() */
caaf5a
58eb1Ø #include "mempool.h"
d1af5a
e738e5 /*
9e9985 ·* The memory pool allocation functions
6e775e ·*
358154 ·* These are based on a linked list of memory blocks, usually of uniform
5dbfdd ·* size. ·New memory is allocated from the tail of the current block,
Ø941cf ·* until that is inadequate, then a new block is allocated.
e85b44 ·* The entire pool can be freed at once by calling memPoolFree().
5e495d ·*/
447186 struct PoolBuf {
d44eØØ ▷       struct PoolBuf *next;
4c56e8 ▷       unsigned size;
5dda85 ▷       /* Data follows */
8182f7 };
aaaf5a
734ØØb /* The prototype empty pool, including the default allocation size. */
a79Øe5 static struct MemPool EmptyPool = { Ø, Ø, Ø, 4Ø96, Ø , Ø, Ø};
f6af5a
a13aee /* Initialize the pool for first use */
d3314a void
6a28aa memPoolInit(struct MemPool *pool)
56bb36 {
Øe9d8f ▷       *pool = EmptyPool;
2fefe6 }
ffaf5a
62ebØ7 /* Set the pool's purge function */
7Ø314a void
5Ø3928 memPoolSetPurge(struct MemPool *pool, int (*purge)(void *), void *arg)
7dbb36 {
6a533c ▷       pool->purge = purge;
d37622 ▷       pool->purgearg = arg;
ecefe6 }
75af5a
247723 /* Free all the memory in the pool */
51314a void
dcd398 memPoolEmpty(struct MemPool *pool)
84bb36 {
deaØ3d ▷       struct PoolBuf *buf;
Ø9af5a
b1b1f9 ▷       while ((buf = pool->head) != Ø) {
644b2a ▷       ▷       pool->head = buf->next;
469151 ▷       ▷       free(buf);
5e1aea ▷       }
2689e2 ▷       pool->freespace = Ø;
e5c46f ▷       pool->totalsize = Ø;
3befe6 }
4eaf5a
e8af5a
8938e5 /*
7e6c3c ·* Restore a pool to a marked position, freeing subsequently allocated
c8e68e ·* memory.
Ø3495d ·*/
Ø2314a void
7deØ28 memPoolCutBack(struct MemPool *pool, struct MemPool const *cutback)
c7bb36 {
d2aØ3d ▷       struct PoolBuf *buf;
1aaf5a
c829e5 ▷       assert(pool);
27e87c ▷       assert(cutback);
bf7dc6 ▷       assert(pool->totalsize >= cutback->totalsize);
83af5a
```

```
3446e7 ▷       while((buf = pool->head) != cutback->head) {
Ø84b2a ▷       ▷       pool->head = buf->next;
bc9151 ▷       ▷       free(buf);
421aea ▷       }
e5a8Ø5 ▷       *pool = *cutback;
f5efe6 }
a3af5a
a338e5 /*
eØc88e ‘* Allocate a chunk of memory for a structure. ‘Alignment is assumed to be
94db5a ‘* a power of 2. ‘It could be generalized, if that ever becomes relevant.
e4c2bd ‘* Note that alignment is from the beginning of an allocated chunk, which
2e72Ø4 ‘* is guaranteed by ANSI to be as aligned as can possibly matter.
ff495d ‘*/
b69a44 void *
8ffd49 memPoolAlloc(struct MemPool *pool, unsigned len, unsigned alignment)
a5bb36 {
7be4bd ▷       char *p;
3b95d1 ▷       unsigned t;
88af5a
43b38c ▷       /* Where to allocate next object */
24161a ▷       p = pool->freeptr;
6745fd ▷       /* How far it is from the beginning of the chunk. */
33ØØØ4 ▷       t = p - (char *)pool->head;
af2ef4 ▷       /* How much to round up freeptr to make alignment */
f8bfae ▷       t = -t & --alignment;
29af5a
cf85c7 ▷       /* Okay, does it fit? */
f9fa7a ▷       if (pool->freespace >= len+t) {
adbdfc ▷       ▷       pool->freespace -= len+t;
c6b7e6 ▷       ▷       p += t;
d5638f ▷       ▷       pool->freeptr = p + len;
6a9bØØ ▷       ▷       return p;
4e1aea ▷       }
b8af5a
Ø9541c ▷       /* It does not fit in the current chunk. ‘Go for a bigger chunk. */
Ø4af5a
48bØba ▷       /* First, figure out how much to skip at the beginning of the chunk */
1bdd74 ▷       alignment &= -(unsigned)sizeof(struct PoolBuf);
dØ935Ø ▷       alignment += sizeof(struct PoolBuf);
66ae62 ▷       /* Then, figure out a chunk size that will fit */
a7e4Øf ▷       t = pool->chunksize;
9Ø4aaØ ▷       assert(t);
e3aØc5 ▷       while (len + alignment > t)
3a75bc ▷       ▷       t *= 2;
bdbb77 ▷       while ((p = malloc(t)) == Ø) {
586364 ▷       ▷       /* If that didn’t work, try purging or smaller allocations */
bØd4ea ▷       ▷       if (!pool->purge ¦¦ !pool->purge(pool->purgearg)) {
2b336f ▷       ▷       ▷       t /= 2;
51f8e4 ▷       ▷       ▷       if (len + alignment > t)
bfb1c1 ▷       ▷       ▷       ▷       fputs("Out of memory!\n", stderr);
7dd6c4 ▷       ▷       ▷       ▷       exit (1);▷      /* Failed */
95467a ▷       ▷       }
6f1aea ▷       }
74af5a
ee96de ▷       /* Update the various pointers. */
5f867b ▷       pool->totalsize += t;
5Øa399 ▷       ((struct PoolBuf *)p)->next = pool->head;
91b6cd ▷       ((struct PoolBuf *)p)->size = t;
1ce5d4 ▷       pool->head = (struct PoolBuf *)p;
aØ97af ▷       pool->freespace = t - len - alignment;
614b24 ▷       p += alignment;
Ød7Ø7b ▷       pool->freeptr = p + len;
44af5a
abc8cf ▷       return p;
9fefe6 }
```

```
--a931 ØØ1581e675e8ØØ2ØØØ9 Page 1 of mempool.h

d29399 /* $Id: mempool.h,v 1.2 1997/11/13 23:53:Ø9 colin Exp $ */
6daf5a
fbb718 #ifndef MEMPOOL␣H
7Ø6a41 #define MEMPOOL␣H
a4af5a
8e853a typedef struct MemPool {
7be9ad ▷       struct PoolBuf *head;
3eb8Øf ▷       char *freeptr;
6211fe ▷       unsigned freespace;
8Øfec9 ▷       unsigned chunksize;▷    /* Default starting point */
27699a ▷       unsigned long totalsize;
ff4f9d ▷       int (*purge)(void *);▷  /* Return non-zero to retry alloc */
cØ66c3 ▷       void *purgearg;
af9d87 } MemPool;
cbaf5a
4c885d /* A global pool for miscellaneous stuff. */
317b5e extern struct MemPool MiscPool;
beaf5a
8Ø38e5 /*
Ø9f456 ‘* Nice clean interfaces
7b495d ‘*/
571ac4 void memPoolInit(struct MemPool *pool);
dc159e void memPoolSetPurge(struct MemPool *pool, int (*purge)(void *), void *arg);
c546c7 void memPoolEmpty(struct MemPool *pool);
86cff6 void memPoolCutBack(struct MemPool *dest, struct MemPool const *cutback);
2Øb125 void *memPoolAlloc(struct MemPool *pool, unsigned len, unsigned alignment);
7457de #ifdef DEADCODE
b8c693 char const *memPoolStore(struct MemPool *pool, char const *str);
7e7454 #endif
56af5a
b7da72 /* Lookie here! ‘An ASNI-compliant alignment finder! */
57cbc2 #define alignof(type) (sizeof(struct{type ␣x; char ␣y;}) - sizeof(type))
dbaf5a
84e554 #define memPoolNew(pool, type) memPoolAlloc(pool, sizeof(type), alignof(type))
Ø6af5a
58c55Ø #endif /* MEMPOOL␣H */
```

```
bc38e5 /*
86f6Ø1 ‘* util.c -- Miscellaneous shared code/data
b1775e ‘*
82a5ec ‘* Copyright (C) 1997 Pretty Good Privacy, Inc.
82775e ‘*
1d7659 ‘* Written by Mark H. Weaver
f6775e ‘*
45c54Ø ‘* $Id: util.c,v 1.11 1997/11/Ø7 ØØ:44:1Ø mhw Exp $
31495d ‘*/
e7af5a
eebea3 #include <stdlib.h>
Ød49Øa #include "util.h"
94af5a
764Ø48 char const hexDigits[] = "Ø123456789abcdef";
231ad8 char const radix64Digits[] =
26bb59 #if Ø▷  /* Standard */
df22a4 ▷   "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyzØ123456789+/";
382d16 #else▷  /* Modified form that avoids hard-to-OCR characters */
a37884 ▷   "ABCDEFGHIJKLMNPQRSTVWXYZabcdehijklmnpqtuwy145689\\^!#$%&*+=/:<>?@";
827454 #endif
eaaf5a
7f879a signed char hexDigitsInv[256];
ea1efØ signed char radix64DigitsInv[256];
cfaf5a
cØea65 /* teun: moved intitialisation of all three CRCPoly’s to initUtil() */
fcaf5a
6f3dce /* CRC-CCITT: x^16 + x^12 + x^5 + 1 */
d16497 CRCPoly▷crcCCITTPoly;
8a38e5 /*
b3ØeaØ ‘* PRZ’s magic 24-bit polynomial - (x+1) * (irreducible of degree 23)
Ø17ea3 ‘* x^24 +x^23 +x^18 +x^17 +x^14 +x^11 +x^1Ø +x^7 +x^6 +x^5 +x^4 +x^3 +x +1
2fb91c ‘* (Developed by Neal Glover). ‘Note: this is bit-reversed from the form
1bd814 ‘* used in PGP, Øx1864cfb.
Øc495d ‘*/
335f89 CRCPoly▷crc24Poly;
7426Ø7 /* CRC-32: x^32+x^26+x^23+x^22+x^16+x^12+x^11+x^1Ø+x^8+x^7+x^5+x^4+x^2+x+1 */
9b86fb CRCPoly▷crc32Poly;
98af5a
Ø19Ød3 EncodeFormat const▷ hexFormat =
46bb36 {
b5dde8 ▷   NULL,▷  ▷   ▷   ▷   /* nextFormat */
cc2d65 ▷   ’-’,▷   ▷   ▷   ▷   /* headerTypeChar */
848ec8 ▷   hexDigits,▷ ▷   ▷   /* digits */
153Ø9c ▷   hexDigitsInv,▷  ▷   /* digitsInv */
8f92Øa ▷   4,▷ ▷   ▷   ▷   ▷   /* bitsPerDigit */
34Ø8c5 ▷   16,▷▷   ▷   ▷   ▷   /* radix */
98e88e ▷   &crcCCITTPoly,▷ ▷   /* lineCRC */
cf7d2f ▷   &crc32Poly,▷▷   ▷   /* pageCRC */
6ØaØ91 ▷   8,▷ ▷   ▷   ▷   ▷   /* runningCRCBits */
c71574 ▷   24,▷▷   ▷   ▷   ▷   /* runningCRCShift */
9ca777 ▷   ØxFF▷   ▷   ▷   ▷   /* runningCRCMask */
6182f7 };
ebaf5a
45922e EncodeFormat const▷ radix64Format =
89bb36 {
87997f ▷   &hexFormat,▷▷   ▷   /* nextFormat */
443dbf ▷   ’A’,▷   ▷   ▷   ▷   /* headerTypeChar */
7d69e5 ▷   radix64Digits,▷ ▷   /* digits */
79f9Øc ▷   radix64DigitsInv,▷  /* digitsInv */
Ø9Ø7d6 ▷   6,▷ ▷   ▷   ▷   ▷   /* bitsPerDigit */
99c683 ▷   64,▷▷   ▷   ▷   ▷   /* radix */
7Ø93fc ▷   &crc24Poly,▷▷   ▷   /* lineCRC */
aØ7d2f ▷   &crc32Poly,▷▷   ▷   /* pageCRC */
192e45 ▷   12,▷▷   ▷   ▷   ▷   /* runningCRCBits */
e67eØa ▷   2Ø,▷▷   ▷   ▷   ▷   /* runningCRCShift */
d635e5 ▷   ØxFFF▷  ▷   ▷   ▷   /* runningCRCMask */
2Ø82f7 };
58af5a
Øb4a51 EncodeFormat const *▷   firstFormat = &radix64Format;
acaf5a
2faf5a
7c3eff static void InitCRCPoly(CRCPoly *poly)
```

```
2cbb36 {
39aa8b ▷   int▷▷   i, oneBit;
72299c ▷   CRC▷▷   crc = 1;
bØaf5a
38cØeØ ▷   poly->table[Ø] = Ø;
8ec7af ▷   for (oneBit = Øx8Ø; oneBit > Ø; oneBit >>= 1) {
8Ø1d71 ▷   ▷   crc = (crc >> 1) ^ ((crc & 1) ? poly->poly : Ø);
41c317 ▷   ▷   for (i = Ø; i < Øx1ØØ; i += 2 * oneBit)
9c2487 ▷   ▷   ▷   poly->table[i + oneBit] = poly->table[i] ^ crc;
c69371 ▷   }
7befe6 }
8Øaf5a
56df6f CRC CalculateCRC(CRCPoly const *poly, CRC crc,
Øfca8e ▷   ▷   ▷   ▷   ‘byte const *buffer, size␣t length)
65bb36 {
ca49f8 ▷   while (length--)
Ø18edf ▷   ▷   crc = (crc >> 8) ^ poly->table[(crc & ØxFF) ^ (*buffer++)];
4dba79 ▷   return crc;
c2efe6 }
31af5a
6d7f13 CRC ReverseCRC(CRCPoly const *poly, CRC crc, byte b)
6Øbb36 {
1a2Ød7 ▷   int▷▷   i, highBit = poly->highBit;
1caf5a
df1246 ▷   for (i = Ø; i < 8; i++) {
254e64 ▷   ▷   if (crc & highBit)▷ ▷   /* highBit is 2^(poly->bits-1) */
4d32eb ▷   ▷   ▷   crc = ((crc ^ poly->poly) << 1) ^ 1;
41Øe1d ▷   ▷   else
d4f893 ▷   ▷   ▷   crc <<= 1;
e99371 ▷   }
6aafØf ▷   return crc ^ b;
ffefe6 }
d5af5a
8ef81e static void InitDigitsInv(char const *digits, signed char *digitsInv)
75bb36 {
c4e595 ▷   int▷▷   i;
2baf5a
3bf151 ▷   for (i = Ø; i < 256; i++)
7e7659 ▷   ▷   digitsInv[i] = -1;
23be35 ▷   for (i = Ø; digits[i]; i++)
facceb ▷   ▷   digitsInv[(byte)digits[i]] = i;
22efe6 }
efaf5a
a613dØ /* Returns the number of chars encoded */
a64b67 int EncodeCheckDigits(EncodeFormat const *fmt, word32 num,
bfc4ee ▷   ▷   ▷   ▷   ▷   ‘‘int numBits, char *dest)
2abb36 {
b1da2c ▷   int▷▷   destLen = EncodedLength(fmt, numBits);
7decdØ ▷   word32▷ digitMask = fmt->radix - 1;
4de595 ▷   int▷▷   i;
1eaf5a
cØad89 ▷   for (i = destLen - 1; i >= Ø; i--)
1ec7a1 ▷   {
158a17 ▷   ▷   dest[i] = EncodeDigit(fmt, num & digitMask);
e14Øb5 ▷   ▷   num >>= fmt->bitsPerDigit;
a59371 ▷   }
911b14 ▷   return destLen;
7cefe6 }
45af5a
aØd352 /* Returns 1 if there’s an error */
f9f54b int DecodeCheckDigits(EncodeFormat const *fmt, char const *src, char **endPtr,
646b86 ▷   ▷   ▷   ▷   ▷   ‘‘int numBits, word32 *valuePtr)
9abb36 {
d72485 ▷   word32▷ value = Ø;
df644b ▷   int▷▷   digitValue;
772325 ▷   int▷▷   i = EncodedLength(fmt, numBits);
f6af5a
9Ø5b55 ▷   while (i--)
14c7a1 ▷   {
d8fba1 ▷   ▷   digitValue = DecodeDigit(fmt, *src++);
4aØ36c ▷   ▷   if (digitValue < Ø)
4c5b34 ▷   ▷   {
```

```
7f289a ▷   ▷   ▷   /* Invalid digit found */
da659c ▷   ▷   ▷   *valuePtr = Ø;
3e38e3 ▷   ▷   ▷   if (endPtr)
628972 ▷   ▷   ▷   ▷   *endPtr = NULL;
29db98 ▷   ▷   ▷   return 1;
1ØØfe4 ▷   ▷   }
74d35f ▷   ▷   value = (value << fmt->bitsPerDigit) ¦ digitValue;
Ø29371 ▷   }
442aff ▷   *valuePtr = value;
83d564 ▷   if (endPtr)
4ece2d ▷   ▷   *endPtr = (char *)src;
9f3e6b ▷   return Ø;
8Øefe6 }
fdaf5a
3b9ce8 EncodeFormat const *FindFormat(char headerTypeChar)
a1bb36 {
d96176 ▷   EncodeFormat const *▷   fmt = firstFormat;
88af5a
Ø61Ø7b ▷   while (fmt && fmt->headerTypeChar != headerTypeChar)
35b95f ▷   ▷   fmt = fmt->nextFormat;
4be4e5 ▷   return fmt;
Ø5efe6 }
dfaf5a
877925 void InitUtil()
88bb36 {
866733 ▷   /* teun: removed "{ }" for MS VC compile */
dfaf5a
95632Ø ▷   crcCCITTPoly.bits = 16;
36ae0a ▷   crcCCITTPoly.poly = Øx84Ø8;
57e3Ø2 ▷   crcCCITTPoly.highBit = Øx8ØØØ;
2aaf5a
71bd8c ▷   crc24Poly.bits = 24;
f9a8b3 ▷   crc24Poly.poly = Øxdf3261;
bb945f ▷   crc24Poly.highBit = Øx8ØØØØØ;
e1af5a
a3ef2b ▷   crc32Poly.bits = 32;
d1d27f ▷   crc32Poly.poly = Øxedb8832Ø;
3aØ754 ▷   crc32Poly.highBit = Øx8ØØØØØØØ;
5daf5a
23fe2e ▷   InitCRCPoly(&crcCCITTPoly);
2b8228 ▷   InitCRCPoly(&crc24Poly);
3Ød264 ▷   InitCRCPoly(&crc32Poly);
Ø7ca42 ▷   InitDigitsInv(hexDigits, hexDigitsInv);
75ce19 ▷   InitDigitsInv(radix64Digits, radix64DigitsInv);
84efe6 }
c9af5a
c5af5a
8938e5 /*
7ce6c5 ·* Local Variables:
4c9b19 ·* tab-width: 4
5Øe7a4 ·* End:
f2b612 ·* vi: ts=4 sw=4
986c42 ·* vim: si
6Ø495d ·*/
```

```
bc38e5 /*
a5e984 ·* util.h -- Miscellaneous defines
Ø1775e ·*
8ba5ec ·* Copyright (C) 1997 Pretty Good Privacy, Inc.
2e775e ·*
e77659 ·* Written by Mark H. Weaver
b4775e ·*
b34b9a ·* $Id: util.h,v 1.23 1997/11/12 23:28:56 mhw Exp $
3d495d ·*/
78af5a
6466b6 #ifndef UTIL_H
2f9484 #define UTIL_H 1
52af5a
cd9117 typedef unsigned long▷  word32;
d614b1 typedef unsigned short▷ word16;
99Ø2Øc typedef unsigned char▷  byte;
Ø6af5a
Ø29e6e #define FMT32▷  "%Ø8lx"
5919d3 #define FMT16▷  "%Ø4x"
af87e6 #define FMT8▷   "%Ø2x"
edaf5a
2a278f #define TAB_CHAR▷   ▷   '\244'▷ /* Currency symbol, like o in top of x */
b61cb8 #define TAB_STRING▷ ▷   "\244"
8af2d8 #define TAB_PAD_CHAR▷   ' '▷▷   /* The fact that this is space has leaked. */
b9Øde1 #define TAB_PAD_STRING▷ " "▷▷   /* It may not be freely changed. */
8cd64f #define FORMFEED_CHAR▷  '\245'▷ /* Yen symbol, like = on top of Y */
99c6c3 #define FORMFEED_STRING▷"\245"
9367a6 #define SPACE_CHAR▷ ▷   '\267'▷ /* Middle dot, or bullet */
622cb7 #define SPACE_STRING▷   "\267"
454439 #define CONTIN_CHAR▷▷   '\266'▷ /* Pilcrow (paragraph symbol) */
54b2e9 #define CONTIN_STRING▷  "\266"
83af5a
47e3b5 #define BYTES_PER_LINE▷ 6Ø▷ ▷   /* When using radix 64 */
99af5a
1Øcd1Ø #define LINES_PER_PAGE▷ 72▷ ▷   /* Exclusive of 2 header lines */
Ød32b9 #define LINE_LENGTH▷▷   8Ø
d8bb43 #define PREFIX_LENGTH▷  7▷  ▷   /* Length of prefix, including the space */
b7af5a
b33a32 #define HDR_PREFIX_CHAR▷▷   '-'
334Ø5a #define RADIX64_END_CHAR▷   '-'
66af5a
e49Ø91 typedef struct EncodeFormat▷▷   EncodeFormat;
d19f11 typedef word32▷ ▷   ▷   ▷   ▷   CRC;
4fc357 typedef word16▷ ▷   ▷   ▷   ▷   CRCFragment;
7baf5a
fd1741 typedef struct
46bb36 {
fdd11b ▷   CRC▷▷   ▷   table[256];
be337Ø ▷   int▷▷   ▷   bits;
9Ø753f ▷   CRC▷▷   ▷   poly;
b4Ø9b4 ▷   CRC▷▷   ▷   highBit;
b14b93 } CRCPoly;
4baf5a
e4c568 struct EncodeFormat
45bb36 {
9d4c38 ▷   EncodeFormat const *nextFormat;
f7ØØ1d ▷   char▷   ▷   ▷   ▷   headerTypeChar;
9a475d ▷   char const *▷   ▷   digits;
7aba7f ▷   signed char const *▷digitsInv;
b645b3 ▷   int▷▷   ▷   ▷   ▷   bitsPerDigit;
f87c33 ▷   int▷▷   ▷   ▷   ▷   radix;
baeb86 ▷   CRCPoly const *▷▷   lineCRC;
db9658 ▷   CRCPoly▷const *▷▷   pageCRC;
Ø8e8b2 ▷   int▷▷   ▷   ▷   ▷   runningCRCBits;
769115 ▷   int▷▷   ▷   ▷   ▷   runningCRCShift;
Øc9b6d ▷   int▷▷   ▷   ▷   ▷   runningCRCMask;
d482f7 };
5daf5a
abaf5a
c6bf51 #define HDR_ENC_LENGTH▷ ▷   19▷ ▷   /* Length of encoded prefix on header */
59af5a
59fc9c #define HDR_VERSION_BITS▷   4
```

```
29a5d5 #define HDR_FLAG_BITS▷  ▷   8
fa92ac /* Page CRC bits omitted, since it's not constant */
ØØf07a #define HDR_TABWIDTH_BITS▷  4
273e3Ø #define HDR_PRODNUM_BITS▷   12
e5eØ25 #define HDR_FILENUM_BITS▷   16
7aaf5a
4caf5a
1f8332 /* Enough to hold one whole page of munged data */
c18Ø44 /* There is no point making this excessively too large */
ec29f2 #define PAGE_BUFFER_SIZE▷   8192
bdaf5a
12b97c #if PAGE_BUFFER_SIZE < (LINES_PER_PAGE + 2) * (LINE_LENGTH + PREFIX_LENGTH + 2)
f1bd5c #error PAGE_BUFFER_SIZE is too small
e97454 #endif
c5af5a
c4af5a
3ad8fa /* Header flags */
ca7882 #define HDR_FLAG_LASTPAGE▷  ØxØ1▷   /* Indicates last page of file */
d4af5a
d4af5a
b11555 #define elemsof(array) (sizeof(array)/sizeof(*(array)))
2aaf5a
82af5a
215a96 extern char const▷  hexDigits[];
36bfa2 extern char const▷  radix64Digits[];
e1af5a
3ed751 extern signed char▷ hexDigitsInv[256];
a166df extern signed char▷ radix64DigitsInv[256];
64af5a
54538b extern CRCPoly▷ ▷   crcCCITTPoly, crc24Poly, crc32Poly;
Ø2af5a
1b77cd extern EncodeFormat const▷  ▷   hexFormat, radix64Format;
29dcfa extern EncodeFormat const *▷▷   firstFormat;
fdaf5a
6baf5a
cdØcbc #define HexDigitValue(ch)▷  ▷   hexDigitsInv[(byte)(ch)]
3b61ed #define Radix64DigitValue(ch)▷  radix64DigitsInv[(byte)(ch)]
96af5a
9418f1 /* Returns the number of chars needed to encode the given number of bits */
bde963 #define EncodedLength(fmt, numBits)▷\
96e65f ▷   ▷   (((numBits) + (fmt)->bitsPerDigit - 1) / (fmt)->bitsPerDigit)
5d328Ø #define EncodeDigit(fmt, value)▷▷   ((fmt)->digits[value])
a6ff4d #define DecodeDigit(fmt, digit)▷▷   ((fmt)->digitsInv[(byte)digit])
6faf5a
3f796f #define AdvanceCRC(poly, crc, b)▷   \
dØ54c8 ▷   ▷   ((crc) >> 8) ^ (poly)->table[((crc) ^ (b)) & ØxFF]
a1af5a
2c64db #define RunningCRCFromPageCRC(fmt, pageCRC)▷\
a772c9 ▷   ▷   (((pageCRC) >> (fmt)->runningCRCShift) & (fmt)->runningCRCMask)
Ødaf5a
f5af5a
4fdf6f CRC CalculateCRC(CRCPoly const *poly, CRC crc,
ØØ6d9c ▷   ▷   ▷   ▷   ·byte const *buffer, size_t length);
4f2245 CRC ReverseCRC(CRCPoly const *poly, CRC crc, byte b);
1Øaf5a
6713dØ /* Returns the number of chars encoded */
c54b67 int EncodeCheckDigits(EncodeFormat const *fmt, word32 num,
f1Øe94 ▷   ▷   ▷   ▷   ▷   ··int numBits, char *dest);
9baf5a
eed352 /* Returns 1 if there's an error */
62f54b int DecodeCheckDigits(EncodeFormat const *fmt, char const *src, char **endPtr,
18e175 ▷   ▷   ▷   ▷   ▷   ··int numBits, word32 *valuePtr);
8aaf5a
386bfa EncodeFormat const *FindFormat(char headerTypeChar);
Øcaf5a
cd76f6 void InitUtil();
e1af5a
75af5a
d3d5be #endif /* !UTIL_H */
86af5a
5Ø38e5 /*
4ee6c5 ·* Local Variables:
```

```
e89b19 ‘* tab-width: 4
69e7a4 ‘* End:
2bb612 ‘* vi: ts=4 sw=4
2a6c42 ‘* vim: si
69495d ‘*/
```

```
bc38e5 /*
1479cc ·* repair.c -- Program which reconstructs scanned source, locates errors,
f1e867 ·*▷ ▷   ▷   ···and tries to fix most of them automatically. ·If it
9de543 ·* ···········can't, it drops you into an editor on the appropriate
cc86e3 ·* ···········line for manual correction.
ae775e ·*
c3ee77 ·* Given a file "foo", this appends corrected output to "foo.out"
5b8d1e ·* and copies remaining uncorrected input in "foo.in". ·If "foo.in"
99c1ae ·* exists initially, "foo" is ignored and only "foo.in" is processed.
51db5c ·* Thus, re-running it repeatedly, possibly with other correction
9fb1ae ·* techniques in between, will result in correct output in "foo.out"
66d449 ·* and an empty "foo.in" file.
6f775e ·*
77adb7 ·* This can automatically invoke an editor for you on the .in file
eebØ38 ·* and re-run itself. ·The editor is chosen in the first available way:
78465c ·* - The -e command-line argument takes a printf() format string to
63d16a ·* ··format the editor invocation command line with the line number and
581ddØ ·* ··filename. ·E.g. "emacs +%u %s". ·%u and %s must appear, in that order.
e25fd8 ·* - Failing that, the default is "$VISUAL +%u %s"
cfa4b1 ·* - Failing that, the default is "$EDITOR +%u %s"
37132f ·* - Failing that, the program prints the error location and exits.
c7852f ·* ··Specifying -e- forces this behaviour.
Øb775e ·*
f8a5ec ·* Copyright (C) 1997 Pretty Good Privacy, Inc.
ab775e ·*
56bd99 ·* Designed by Colin Plumb, Mark H. Weaver, and Philip R. Zimmermann
f7fbf7 ·* Written by Colin Plumb
b8775e ·*
158237 ·* $Id: repair.c,v 1.37 1997/11/14 Ø8:39:4Ø mhw Exp $
58495d ·*/
bcaf5a
Øabb5f #include <assert.h>
54feb2 #include <stdio.h>
b9324c #include <string.h>
a5b1cb #include <ctype.h>
b3495d #include <errno.h>
2cb16e #include <signal.h>
eØaf5a
fe49Øa #include "util.h"
Ø46Ø9b #include "heap.h"
53eb1Ø #include "mempool.h"
fØdØ4a #include "subst.h"
b9af5a
2938e5 /*
c16222 ·* The internal form of a substitution. ·These are stored on
7f3fbc ·* lists indexed by the first character of the input substitution.
c6495d ·*/
23fb7d typedef struct Substitution {
d813fe ▷   struct Substitution *next;
abbc2e ▷   char const *input, *output;
e94c7f ▷   size_t inlen, outlen;
598ec8 ▷   HeapCost cost, cost2;
db7725 ▷   FilterFunc *filter;
d5Ø8ca ▷   unsigned int index;▷/* Consecutive serial numbers */
68b5Øc } Substitution;
68af5a
62Ø9ba struct Substitution const substNull = { NULL, "", "", Ø, Ø, Ø, Ø, Ø };
61af5a
9Ø38e5 /*
674c9b ·* This might get increased later to support multiple classes of
736Ø59 ·* substitutions, for different contexts. ·Currently, only one
6Ø579f ·* is used.
75495d ·*/
c6bf86 #define SUBST_CLASSES 1
23af5a
1e5dbc /* List of substitutions, indexed by first character, plus a catch-all */
8e8f9c Substitution *substitutions[SUBST_CLASSES][Øx1Ø1];
9Øaf5a
4c38e5 /*
9e78c5 ·* The pool of Substitution structures. ·Remains alive for the entire
541dcf ·* execution of the program.
7b495d ·*/
```

```
4c542a static MemPool substPool;
693Ø53 static Substitution *substFree;
2fdØ9Ø static unsigned int substCount = 1;▷/* Preallcoate Ø to substNull */
6Ød161 static unsigned int substFirstDynamic;
61b8Ø6 #define SubstIsDynamic(s) ((s)->index >= substFirstDynamic)
ecf319 /* Adjust the substitution based on noccurrences this page */
2a32f4 #define SubstAdjust(s,n) ((s)->cost = (s)->cost2)
442657 /* Is this a nasty-line substitution? */
e2441Ø #define SubstIsNasty(s) ((s)->cost2 == COST_INFINITY)
8aaf5a
74c88d /* Every possible single-character string */
1af27e static char substChars[512];
d45596 #define SubstString(c) (substChars+2*((c)&255))
6Øaf5a
9e41a9 /* Set the list of substitutions to empty */
6b2978 static void
e85fdb SubstInit(void)
42bb36 {
791b5c ▷   unsigned int i, j;
cØaf5a
6865eØ ▷   memPoolInit(&substPool);
31c567 ▷   substFree = Ø;
fc9b5f ▷   substCount = 1;▷/* Number zero is reserved for uncounted substitutions */
884cbe ▷   for (i = Ø; i < elemsof(substitutions); i++)
41a7fb ▷   ▷   for (j = Ø; j < elemsof(*substitutions); j++)
b7d311 ▷   ▷   ▷   substitutions[i][j] = NULL;
Ø4af5a
1Øe75e ▷   for (i = Ø; i < 256; i++) {
431226 ▷   ▷   substChars[2*i] = (char)i;
841f5a ▷   ▷   substChars[2*i+1] = Ø;
Øb9371 ▷   }
f3efe6 }
e6af5a
bc38e5 /*
1997Ø1 ·* For dynamically allocated substitutions, we maintain a free list.
d19377 ·* Each substitution has a unique serial number. ·These are retained
75Ø24Ø ·* if a substitution goes on the free list, to keep substCount from
7eØ336 ·* ratcheting upwards indefinitely while still guaranteeing uniqueness.
5Ø495d ·*/
45ed99 static Substitution *
Øebd82 SubstAlloc(void)
53bb36 {
c6b1a2 ▷   struct Substitution *subst = substFree;
adaf5a
4f9ce3 ▷   if (subst) {
c1381b ▷   ▷   substFree = subst->next;
7dfØ37 ▷   } else {
1a3e99 ▷   ▷   subst = memPoolNew(&substPool, Substitution);
9Øe87e ▷   ▷   subst->index = substCount++;
979371 ▷   }
e1Ø1ad ▷   return subst;
Ø6efe6 }
9Øaf5a
ba2978 static void
aØfbc2 SubstFree(Substitution *subst)
bbbb36 {
efeØf6 ▷   subst->next = substFree;
2733Øb ▷   substFree = subst;
63efe6 }
c8af5a
8eed99 static Substitution *
c662Ø6 MakeSubst(char const *input, char const *output, HeapCost cost, HeapCost cost2,
eae515 ▷   FilterFunc *filter, int class)
3fbb36 {
72dcØ1 ▷   struct Substitution *subst, **head;
fcaf5a
e64cØ4 ▷   subst = SubstAlloc();
b1Øda5 ▷   subst->input = input;
2e5e9a ▷   subst->output = output;
c931f5 ▷   subst->inlen = strlen(input);
8c4a42 ▷   subst->outlen = strlen(output);
4bf676 ▷   subst->cost = cost;
```

```
8f3d1d ▷   subst->cost2 = cost2;
a6b921 ▷   subst->filter = filter;
e3af5a
Øbd8af ▷   /*
cdba85 ▷   ·* Ignore certain substitutions when printing stats.
fc64b1 ▷   ·* Identity substitutions, and the tab/space tweaking.
28a4e3 ▷   ·*/
ØaØ474 ▷   if (strcmp(input, output) == Ø ¦¦ strcmp(input, TAB_STRING) == Ø ¦¦
dec264 ▷   ▷   (input[Ø] == ' ' && input[1] == Ø && output[Ø] == Ø)) {
9ff78e ▷   ▷   ▷   if (subst->index == substCount-1)
fcacdd ▷   ▷   ▷   ▷   substCount--;
649de6 ▷   ▷   ▷   subst->index = Ø;▷  /* Evil hack */
6a9371 ▷   }
Ø2af5a
174ee4 ▷   head = &substitutions[class][input[class] & 255];
18ae83 ▷   subst->next = *head;
8f8b9c ▷   *head = subst;
58Ø1ad ▷   return subst;
d6efe6 }
afaf5a
be38e5 /*
Øe7544 ·* For each entry in the raw array, turn { "abc", "def", 5" }
5b6Ø78 ·* into cost-5 mappings of "a"->"d", "b"->"e" and "c"->"f".
f99114 ·* If the output string is NULL, the characters are deleted.
bØ4a63 ·* An input string of NULL is the end of table delimiter.
1c495d ·*/
2f2978 static void
555417 SubstSingle(struct RawSubst const *raw, int class)
9Øbb36 {
63bc2e ▷   char const *input, *output;
5cabc3 ▷   int i, o;
4faf5a
e13cb9 ▷   while (raw->input) {
19badc ▷   ▷   input = raw->input;
9b25d9 ▷   ▷   output = raw->output;
e7acd2 ▷   ▷   assert(!output ¦¦ strlen(input) == strlen(output));
61af5a
57d9f1 ▷   ▷   while (*input) {
2cd15f ▷   ▷   ▷   i = *input++;
a2a97a ▷   ▷   ▷   o = output ? *output++ : Ø;
dea3cd ▷   ▷   ▷   (void)MakeSubst(SubstString(i), SubstString(o),
7bcf6Ø ▷   ▷   ▷   ▷   ▷   ▷   ▷   raw->cost, raw->cost2, raw->filter, class);
83Øfe4 ▷   ▷   }
3Øbacb ▷   ▷   raw++;
a69371 ▷   }
c1efe6 }
Øaaf5a
8538e5 /*
c17544 ·* For each entry in the raw array, turn { "abc", "def", 5" }
e1af86 ·* into a cost-5 mappings of "abc"->"def".
764a63 ·* An input string of NULL is the end of table delimiter.
36495d ·*/
be2978 static void
a71ccc SubstMultiple(struct RawSubst const *raw, int class)
7bbb36 {
2b3cb9 ▷   while (raw->input) {
24262f ▷   ▷   (void)MakeSubst(raw->input, raw->output, raw->cost, raw->cost2,
e1295Ø ▷   ▷   ▷   ▷   ▷   ····raw->filter, class);
8fbacb ▷   ▷   raw++;
9f9371 ▷   }
88efe6 }
b3af5a
e97c4e /* Build the substitutions table */
da2978 static void
8474aØ SubstBuild(void)
f2bb36 {
551c36 ▷   SubstInit();
eb6674 ▷   SubstSingle(substSingles, Ø);
1a5785 ▷   SubstMultiple(substMultiples, Ø);
12fc91 ▷   substFirstDynamic = substCount;
9defe6 }
95af5a
```

```
bc38e5 /*
7bØ457 ·* See if the desired substitution already exists
7c495d ·*/
82448Ø static Substitution const *
3e31d3 SubstSearch(char const *in, size_t inlen, char const *out, size_t outlen,
c7e7fc ▷   int class)
82bb36 {
3d4fb3 ▷   Substitution *subst = substitutions[class][in[Ø] & 255];
43af5a
fb43b5 ▷   for (; subst; subst = subst->next) {
e16Ø2d ▷   ▷   if (subst->inlen == inlen && subst->outlen == outlen &&
71f8Øc ▷   ▷   ▷   memcmp(subst->input, in, inlen) == Ø &&
e9Ø97a ▷   ▷   ▷   memcmp(subst->output, out, outlen) == Ø)
32c9a2 ▷   ▷   ▷   ▷   return subst;▷  /* Already exists */
b69371 ▷   }
7a3f59 ▷   return NULL;
e6efe6 }
5faf5a
5daf5a
1e38e5 /*
Ø98eØ1 ·* Create a new dynamic substitution. ·First search to make
925811 ·* sure it doesn't already esist.
Ø4495d ·*/
Ø8448Ø static Substitution const *
1bb434 SubstDynamic(char const *in, char const *out, int class)
Ø1bb36 {
f16aa7 ▷   Substitution const *subst;
5faf5a
a83a34 ▷   subst = SubstSearch(in, strlen(in), out, strlen(out), class);
f28cd1 ▷   return subst ? subst : MakeSubst(in, out, COST_INFINITY,
8bd733 ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ·DYNAMIC_COST_LEARNED, NULL, class);
19efe6 }
e4af5a
1e38e5 /*
cae4bØ ·* Search for the substitution, allocating one if not found.
426c38 ·* the input string is not null-terminated and needs to be copied to
4437c5 ·* an allocated buffer. ·The output string can just be pointer-copied.
2a495d ·*/
21448Ø static Substitution const *
a1e5f6 SubstNasty(char const *in, size_t inlen, char const *out, int class)
85bb36 {
dc6aa7 ▷   Substitution const *subst;
bcaabe ▷   char *string;
98af5a
ØbØ33a ▷   if ((subst = SubstSearch(in, inlen, out, strlen(out), class)) != NULL)
ØØ3b37 ▷   ▷   return subst;
efaf5a
c43ebf ▷   if (!(string = malloc(inlen+1))) {
Ø6bc9a ▷   ▷   fputs("Out of memory!\n", stderr);
ac4bb3 ▷   ▷   exit(1);
fb9371 ▷   }
71add9 ▷   memcpy(string, in, inlen);
afce18 ▷   string[inlen] = Ø;
bØ3dc3 ▷   return MakeSubst(string, out, COST_INFINITY, COST_INFINITY, NULL, class);
e7efe6 }
96af5a
5238e5 /*
513f6Ø ·* The state of the parser.
364Ød Ø ·* Note that this is updated when a ParseNode is *removed* from the heap;
3fc5b1 ·* ParseNodes that are in the heap have ParseStates that reflect the
3Ø8489 ·* state before the substitution has been parsed; this is a copy of the
516579 ·* parents' state, which is after the parsing.
67495d ·*/
dd754b typedef struct ParseState {
7edca9 ▷   CRC page_crc;▷  ▷   ▷   /* Computed per-page CRC */
569221 ▷   word16 flags;▷  ▷   ▷   /* Flags; see below */
58e69c ▷   unsigned char pos;▷ ▷   /* Position on the line */
a89d43 } ParseState;▷  /* 7 bytes, rounded to 8 */
b6af5a
96f645 /* Flags values */
7Ø925a #define PS_MASK_PAGENUM▷ ØxCØØØ /* Digits in header page number (1..3) */
17c9f8 #define PS_SHIFT_PAGENUM▷   ·14▷/* Shift for the above */
```

```
c284ad #define PS_FLAG_EOL      512/* Expect \n next */
8d445f #define PS_FLAG_SPACE    256/* Was last char a space? */
41f6e2 #define PS_FLAG_TAB      128/* Tabbing over a column */
6470de #define PS_FLAG_INHEADER  64/* Current line is a header */
3225b3 #define PS_FLAG_PASTHEADER 32/* A previous line was a header */
496b39 #define PS_FLAG_BINWS     16/* In whitespace after binary data */
1899e9 #define PS_FLAG_BINEND     8/* End of binary data */
ca7b33 #define PS_FLAG_DYNAMIC    4/* Have used ECC this line */
fa3e6e #define PS_MASK_FORMAT     3/* The encoding format (max of 3, for now) */
c1e7b3 #define PS_SHIFT_FORMAT    0/* Shift for the above */
5caf5a
9d78f7 /* Have we started on a second page? Used to force flushing of the first. */
961747 #define InSecondHeader(ps) \
f361f7     ((~(ps)->flags & (PS_FLAG_INHEADER | PS_FLAG_PASTHEADER)) == 0)
5eaf5a
37b710 #define PageNumDigits(pn) (((pn)->ps.flags & PS_MASK_PAGENUM) >> PS_SHIFT_PAGEN
9a1939 UM)
75ddb2 #define PageNumDigitsIncrement(pn) ((pn)->ps.flags += 1<<PS_SHIFT_PAGENUM)
c9af5a
0f1cb6 EncodeFormat const *registeredFormats[4];
17af5a
57fbc2 /* Returns a small integer index */
e0beb7 static int
66b084 registerFormat(EncodeFormat const *format)
7dbb36 {
be4ba4     int i;
c4f382     for (i = 0; i < (int)elemsof(registeredFormats); i++) {
e62ae4         if (registeredFormats[i] == format)
0a9eb9             return i;
2dddd2         if (!registeredFormats[i]) {
4a123e             registeredFormats[i] = format;
519eb9             return i;
2f0fe4         }
0c9371     }
c1dffc     fputs("Registered formats table overflow!\n", stderr);
b4fe22     exit(1);
51efe6 }
06af5a
986d01 #define psFormat(ps) registeredFormats[((ps)->flags & PS_MASK_FORMAT)>>PS_SHIFT
a8fa57 _FORMAT]
0af419 #define pnFormat(pn) psFormat(&(pn)->ps)
c83adc #define psSetFormat(ps, i) \
41be60     ((ps)->flags = ((ps)->flags & ~PS_MASK_FORMAT) | i << PS_SHIFT_FORMAT)
43af5a
0c9bdb typedef struct ParseNode {
7f7e1f     HeapCost cost;
5833b3     unsigned int refcnt;
8f38ad     struct ParseNode *parent;
fe9260     char const *input;
bf19ba     struct Substitution const *subst;
1da55a     struct ParseState ps;
34fb57 } ParseNode;    /* 32 bytes */
bcaf5a
af909b /* A handle for walking backwards through the output stream */
9c236f typedef struct OutputHandle {
15c9a1     ParseNode const *node;
19dbbc     char const *output;
71d55d     unsigned int pos;
155a32 } OutputHandle;
2baf5a
3a6eeb /* Initialize the handle to point to a node (optionally, a position therein) */
352978 static void
e2d078 OutputInit(OutputHandle *oh, ParseNode const *node, char const *p)
a6bb36 {
0dcfb7     oh->node = node;
3caf8e     oh->output = p ? p : node->subst->output + node->subst->outlen;
6e2733     oh->pos = 0;
d0efe6 }
dcaf5a
28d6ae /* Get the *previous* byte */
8ebeb7 static int
7e56f0 OutputGetPrev(OutputHandle *oh)
```

```
2cbb36 {
9Ø7ddb ▷   if (!oh->node)
9aØ43c ▷   ▷   return -1;
a922Øe ▷   for (;;) {
d4d1fØ ▷   ▷   if (oh->output != oh->node->subst->output) {
7aØ37a ▷   ▷   ▷   oh->pos++;
bØbc6e ▷   ▷   ▷   return *--oh->output & 255;
9bØfe4 ▷   ▷   }
f5fb43 ▷   ▷   oh->node = oh->node->parent;
8d4632 ▷   ▷   if (!oh->node)
7dcdcb ▷   ▷   ▷   break;
9Ø4516 ▷   ▷   oh->output = oh->node->subst->output + oh->node->subst->outlen;
ab9371 ▷   }
3fe69f ▷   return -1;
feefe6 }
57af5a
Øb97d7 /* Return the character just before the node - trivial handy wrapper */
edbeb7 static int
84447a OutputPrevChar(ParseNode const *node)
8dbb36 {
5e8bb8 ▷   OutputHandle oh;
89af5a
bd95b1 ▷   OutputInit(&oh, node, NULL);
ccf996 ▷   return OutputGetPrev(&oh);
25efe6 }
d2af5a
ff38e5 /*
72286b ‘* Unget the last retrieved character (and return it), or
c2eØ3c ‘* -1 if that is impossible. ‘At least one character is
272898 ‘* always ungettable, but after that you’re on your own.
23495d ‘*/
d1beb7 static int
5733ad OutputUnget(OutputHandle *oh)
8dbb36 {
625866 ▷   if (oh->node && *oh->output) {
cfa5dØ ▷   ▷   oh->pos--;
918Øae ▷   ▷   return *oh->output++ & 255;
359371 ▷   }
92e69f ▷   return -1;
74efe6 }
98af5a
91caf8 /* The position is useful for comparing two OutputHandles. */
Ø74f85 #define OutputPos(oh) ((oh)->pos)
15af5a
9738e5 /*
e84Ødf ‘* Fill backwards from bufend until you hit the given char.
d37Øe8 ‘* Use -1 to get the whole buffer.
e3495d ‘*/
ffØ83f static char *
54caf1 OutputGetUntil(OutputHandle oh, char *bufend, int end)
8cbb36 {
1138de ▷   int c;
11af5a
d3e787 ▷   while ((c = OutputGetPrev(&oh)) != -1 && c != end)
59872d ▷   ▷   *--bufend = (char)c;
eaed56 ▷   return bufend;
19efe6 }
84af5a
5738e5 /*
d9bce8 ‘* The per-page structure. ‘This is actually global, but describes
Øaf3b2 ‘* the values kept for each page processed.
d1495d ‘*/
89d2d1 typedef struct PerPage {
35c18Ø ▷   CRC page_check;
52311e ▷   char const *maxpos, *minpos;
98Øf42 ▷   unsigned int tabsize;▷  /* Zero means this is a binary page */
55acfØ ▷   unsigned int lines;
4d1dbe ▷   unsigned int retries;▷  /* How many retires since last progress? */
c56d61 ▷   unsigned int max_retries;▷  /* Maximum number of retries needed. */
71db82 } PerPage;
c4af5a
73cddØ PerPage perpage;▷   /* The global */
```

```
e0af5a
752978 static void
e61315 PerPageInit(char const *buf)
6abb36 {
54a482 ▷   perpage.maxpos = perpage.minpos = buf;
31fce9 ▷   perpage.page␣check = Ø;
6Ø0a22e ▷   perpage.tabsize = 4;▷   /* The default */
7dc832 ▷   perpage.lines = perpage.retries = perpage.max␣retries = Ø;
faefe6 }
3aaf5a
a138e5 /*
86540e ‘* Is the tab substitution being looked at acceptable?
6c7ae7 ‘* It is if the length needed to make the tab width come out
e1a3ef ‘* right, it is. ‘Otherwise, it’s junk.
d7495d ‘*/
e38329 HeapCost
73778e TabFilter(struct ParseNode *parent, char const *limit,
44bf9b ▷   struct Substitution const *subst)
a8bb36 {
7a4e7c ▷   int c, tabpos;
9d8bb8 ▷   OutputHandle oh;
98af5a
7fe8a7 ▷   (void)limit;
a44bd4 ▷   if (!perpage.tabsize)
50e847 ▷   ▷   return COST␣INFINITY;▷  /* No interest */
d0af5a
1b598d ▷   /* How wide should the tab be? */
ba0c4a ▷   tabpos = (int)((parent->ps.pos-PREFIX␣LENGTH) % perpage.tabsize);
7c49e6 ▷   if ((int)subst->outlen != (int)perpage.tabsize - tabpos)
ce0283 ▷   ▷   return COST␣INFINITY;
1edde9 ▷   /* The right number - cost if likely, cost2 if unlikely */
cb032f ▷   if (subst->cost == subst->cost2)
ff962b ▷   ▷   return subst->cost;
fe0607 ▷   OutputInit(&oh, parent, NULL);
da43ab ▷   do {
4c6468 ▷   ▷   c = OutputGetPrev(&oh);
a5ff24 ▷   } while (c == ’ ’);
73b939 ▷   return (c == TAB␣CHAR) ? subst->cost : subst->cost2;
5befe6 }
0daf5a
6838e5 /*
2a3119 ‘* Return cost if near blanks (including end-of-line), cost2 if not, and
5a46cb ‘* the average of there is a blank on one side. ‘There are additional
c7b8da ‘* versions for upper- and lower-case. ‘␣ is considered upper-case,
fbd39d ‘* as it’s oftne used in acro identifiers.
6b495d ‘*/
ca8329 HeapCost
6Ø2436 FilterNearBlanks(struct ParseNode *parent, char const *limit,
38bf9b ▷   struct Substitution const *subst)
23bb36 {
21a73f ▷   int c = OutputPrevChar(parent), score = (isspace(c) != Ø);
f2285Ø ▷   char const *p = parent->input + parent->subst->inlen;
78af5a
a3f651 ▷   score += p == limit ¦¦ isspace((unsigned char)*p) != Ø;
0d21af ▷   return (subst->cost*score + subst->cost2*(2-score))/2;
59efe6 }
f0af5a
c38329 HeapCost
b6a5cf FilterNearUpper(struct ParseNode *parent, char const *limit,
d2bf9b ▷   struct Substitution const *subst)
36bb36 {
3b4794 ▷   int c = OutputPrevChar(parent), score = (isupper(c) != Ø ¦¦ c == ’␣’);
635f95 ▷   char const *p = parent->input + subst->inlen;
ecaf5a
bc2434 ▷   score += p != limit && (isupper((unsigned char)*p) != Ø ¦¦ *p == ’␣’);
7521af ▷   return (subst->cost*score + subst->cost2*(2-score))/2;
1defe6 }
caaf5a
bd8329 HeapCost
b661b5 FilterNearXDigit(struct ParseNode *parent, char const *limit,
27bf9b ▷   struct Substitution const *subst)
7cbb36 {
```

```
Øbb55e ▷   int c = OutputPrevChar(parent), score = (isxdigit(c) != Ø);
955f95 ▷   char const *p = parent->input + subst->inlen;
daaf5a
bd4133 ▷   score += p != limit && (isxdigit((unsigned char)*p) != Ø);
f521af ▷   return (subst->cost*score + subst->cost2*(2-score))/2;
43efe6 }
8caf5a
7a8329 HeapCost
a1ab3c FilterNearLower(struct ParseNode *parent, char const *limit,
c2bf9b ▷   struct Substitution const *subst)
bebb36 {
9bb575 ▷   int c = OutputPrevChar(parent), score = (islower(c) != Ø);
da5f95 ▷   char const *p = parent->input + subst->inlen;
28af5a
853d46 ▷   score += p != limit && (islower((unsigned char)*p) != Ø);
e321af ▷   return (subst->cost*score + subst->cost2*(2-score))/2;
33efe6 }
8daf5a
Ø338e5 /*
ed Ø85e ‘* cost2 unless previous character was a space (’ ’ or SPACE_CHAR).
9f6Øed ‘* Note the & 255, necessary since chars might be signed and SPACE_CHAR
7cabd8 ‘* is in the high (negative) half, but c is an int in the range -1..255.
1c495d ‘*/
de8329 HeapCost
Ø34Øb2 FilterFollowsSpace(struct ParseNode *parent, char const *limit,
dbbf9b ▷   struct Substitution const *subst)
18bb36 {
7daf53 ▷   int c = OutputPrevChar(parent);
4ae8a7 ▷   (void)limit;
Ø6436d ▷   return (c == ’ ’ ¦¦ c == (SPACE_CHAR & 255)) ? subst->cost : subst->cost2;
cØefe6 }
7aaf5a
Ø45Øbe /* cost2 unless previous character was duplicate of this one */
798329 HeapCost
1ca5Øf FilterAfterRepeat(struct ParseNode *parent, char const *limit,
Øbbf9b ▷   struct Substitution const *subst)
aabb36 {
f3af53 ▷   int c = OutputPrevChar(parent);
Øae8a7 ▷   (void)limit;
ff5Øb4 ▷   return (c == subst->output[Ø]) ? subst->cost : subst->cost2;
Ø2efe6 }
6baf5a
ce76c6 /* cost2 unless probably the closing quote in a char constant */
558329 HeapCost
Ø65552 FilterCharConst(struct ParseNode *parent, char const *limit,
f8bf9b ▷   struct Substitution const *subst)
bcbb36 {
7b8bb8 ▷   OutputHandle oh;
eØ38de ▷   int c;
4caf5a
78e8a7 ▷   (void)limit;
c1Ø6Ø7 ▷   OutputInit(&oh, parent, NULL);
5Øc936 ▷   c = OutputGetPrev(&oh);
51c936 ▷   c = OutputGetPrev(&oh);
7Ø1Ø7f ▷   if (c == ’\\’)
766468 ▷   ▷   c = OutputGetPrev(&oh);
dØ744b ▷   return (c == ’\’’) ? subst->cost : subst->cost2;
5defe6 }
78af5a
cØ38e5 /*
875ea6 ‘* If the identifier leading up to the current position contains
23b437 ‘* an underscore, then it’s likely the current position is an underscore
ed8df7 ‘* as well; return cost. ‘If it does not, it’s less likely; return cost2.
4c495d ‘*/
cc8329 HeapCost
6dbØfa FilterLikelyUnderscore(struct ParseNode *parent, char const *limit,
6ebf9b ▷   struct Substitution const *subst)
96bb36 {
Øb8bb8 ▷   OutputHandle oh;
fe38de ▷   int c;
a2af5a
49e8a7 ▷   (void)limit;
```

```
b4Ø6Ø7 ▷   OutputInit(&oh, parent, NULL);
c122Øe ▷   for (;;) {
5c6468 ▷   ▷   c = OutputGetPrev(&oh);
adc1ba ▷   ▷   if (c == '␣')
423b75 ▷   ▷   ▷   return subst->cost;
bØ25fe ▷   ▷   if (!isalnum(c))
58b82f ▷   ▷   ▷   return subst->cost2;
f49371 ▷   }
35efe6 }
ceaf5a
87151c /* cost2 unless the following chars seem to be a checksum */
ae8329 HeapCost
7f2f99 FilterChecksumFollows(struct ParseNode *parent, char const *limit,
d9bf9b ▷   struct Substitution const *subst)
e1bb36 {
d6a37f ▷   int i, score = Ø;
8f5f95 ▷   char const *p = parent->input + subst->inlen;
e7af5a
86fad2 ▷   if (limit - p < PREFIX␣LENGTH)
Ø8Ø379 ▷   ▷   return subst->cost2;
91f57c ▷   if (!isspace((unsigned char)p[PREFIX␣LENGTH-1]))
17Ø379 ▷   ▷   return subst->cost2;
f96e19 ▷   for (i = Ø; i < PREFIX␣LENGTH-1; i++)
Ø74Øe9 ▷   ▷   score += (p[i] >= 'Ø' && p[i] <= '9') + (p[i] >= 'a' && p[i] <= 'f');
e4ff7d ▷   i = (score >= PREFIX␣LENGTH-2 ? subst->cost : subst->cost2);
abb739 ▷   /* Magic, since this function is perfect on binary files */
2Øec79 ▷   if (i < COST␣INFINITY && perpage.tabsize == Ø)
Ød9ed6 ▷   ▷   i = Ø;
da2196 ▷   return i;
8Øefe6 }
d4af5a
d24596 /* Manage a *big* pool of ParseNodes */
69af5a
8682Ø6 struct MemPool nodePool;
de1ae2 struct ParseNode *nodeFreeList = Ø;
1Øaf5a
7Øf5f7 /* Prepare for node allocations */
Ø62978 static void
86d95d NodePoolInit(void)
2abb36 {
ff7b45 ▷   memPoolInit(&nodePool);
d52327 ▷   nodeFreeList = NULL;
f9efe6 }
94af5a
9Ø23c3 /* Free all nodes in one swell foop */
7e2978 static void
1374d6 NodePoolCleanup(void)
dØbb36 {
922327 ▷   nodeFreeList = NULL;
5a3d7d ▷   memPoolEmpty(&nodePool);
Ø6efe6 }
26af5a
bde994 /* Allcoate a new (uninitialized) node */
ØØ6b7b static struct ParseNode *
e84e56 NodeAlloc(void)
87bb36 {
3Ø2829 ▷   struct ParseNode *node;
71af5a
bab2f4 ▷   node = nodeFreeList;
3597dd ▷   if (node) {
154775 ▷   ▷   nodeFreeList = node->parent;
f79665 ▷   ▷   return node;
Ø59371 ▷   }
615dc9 ▷   return memPoolNew(&nodePool, ParseNode);
aeefe6 }
bcaf5a
284aa5 /* Free a node for reallocation */
552978 static void
e816ed NodeFree(struct ParseNode *node)
1abb36 {
35e437 ▷   node->parent = nodeFreeList;
37cb9Ø ▷   nodeFreeList = node;
```

```
7aefe6 }
46af5a
2e38e5 /*
a1bede ·* Decrement a node's reference count, freeing it and
bc8bf7 ·* recursively decrementing its parent's if the count
1d68aa ·* goes to zero.
78495d ·*/
252978 static void
bce5bb NodeRelease(struct ParseNode *node)
Ødbb36 {
9d38ad ▷   struct ParseNode *parent;
5382ae ▷   assert(node->refcnt);
54af5a
d1e79f ▷   while (!--node->refcnt) {
eb159a ▷   ▷   parent = node->parent;
beØd8a ▷   ▷   NodeFree(node);
1f9a16 ▷   ▷   if (!parent)
4bcdcb ▷   ▷   ▷   break;
78ed2c ▷   ▷   node = parent;
a39371 ▷   }
Ø8efe6 }
cdaf5a
bbaØØ7 /* Add nodes to the substitution tree */
ØØaf5a
4532Ø2 /* Create a child of the given node, with the given properties. */
6843e9 static ParseNode *
888e1d AddChild(ParseNode *parent, Substitution const *subst, HeapCost cost)
cbbb36 {
dØ9c24 ▷   ParseNode *child;
ccaf5a
aØdØØØ ▷   if (cost == COST_INFINITY)
1dbbde ▷   ▷   return Ø;
41af5a
b9Øbab ▷   cost += parent->cost;
ebbe28 ▷   child = NodeAlloc();
67c392 ▷   *child = *parent;
a3Ø8fØ ▷   /* Child is just like parent, except... */
38c4ca ▷   child->cost = cost;
267fcØ ▷   child->refcnt = 1;▷ /* The heap */
34d2eØ ▷   child->input += subst->inlen;
e8d9ac ▷   child->subst = subst;
3ac973 ▷   child->parent = parent;
633988 ▷   parent->refcnt++;
2f7Ø2a ▷   return child;
66efe6 }
fØaf5a
d99Ø66 /* Hash table of nasty lines, indexed by per-line CRC */
fa869f struct NastyLine {
d398ec ▷   struct NastyLine *next;
18fØ75 ▷   char const *line;
142786 ▷   CRC crc;
cf82f7 };
57af5a
dØed6d #define NASTY_HASH_SIZE 256
6Ø32Øe static struct NastyLine *nastyHash[NASTY_HASH_SIZE];▷   /* All zero */
6aaf5a
626178 struct MemPool nastyStrings, nastyStructs;
ccf341 static CRCPoly const *nastyPoly = &crcCCITTPoly;
b638e5 /*
de8dØa ·* Create a new NastyString entry if it doesn't already exist.
2d454d ·* Note that this expects the string passed to end in a newline which
42518c ·* IS hashed but NOT stored
ab495d ·*/
12b5b4 static struct NastyLine *
5a35eb AddNasty(char const *string)
Ø3bb36 {
3e7b49 ▷   size_t len = strlen(string);▷   /* Including newline */
efbbbf ▷   CRC crc = CalculateCRC(nastyPoly, Ø, (byte const *)string, len);
4dba69 ▷   struct NastyLine *nasty, **nastyp = nastyHash + (crc % NASTY_HASH_SIZE);
5a5ac6 ▷   char *line;
c3af5a
421595 ▷   /* Search for an existing copy */
```

```
5f9657 ▷   while ((nasty = *nastyp) != NULL) {
d4cØ7c ▷   ▷   if (nasty->crc == crc &&
737fcØ ▷   ▷   ▷   memcmp(nasty->line, string, len-1) == Ø &&
bØØ2be ▷   ▷   ▷   nasty->line[len-1] == Ø)
2aab16 ▷   ▷   ▷   ▷   return nasty;
79e9eØ ▷   ▷   nastyp = &nasty->next;
369371 ▷   }
Ø9cØ28 ▷   /* Create a new structure */
ea189e ▷   *nastyp = nasty = memPoolNew(&nastyStructs, struct NastyLine);
a45Ø42 ▷   nasty->next = NULL;
d3363b ▷   nasty->line = line = memPoolAlloc(&nastyStrings, len, 1);
24Ø375 ▷   nasty->crc = crc;
dbd352 ▷   memcpy(line, string, len-1);
61c637 ▷   line[len-1] = Ø;
a5df6d ▷   return nasty;
37efe6 }
27af5a
c82978 static void
6fa8Ø1 RehashNasties(CRCPoly const *poly)
ebbb36 {
Øfdbb8 ▷   struct NastyLine *cur, *head;
672786 ▷   CRC crc;
dc4ba4 ▷   int i;
37f2Øc ▷   size␣t len;
Ø6af5a
5fe9fØ ▷   /* Put everything into one list and clear the hash table */
d3f11b ▷   head = NULL;
adc751 ▷   for (i = Ø; i < (int)elemsof(nastyHash); i++) {
b5125d ▷   ▷   while ((cur = nastyHash[i]) != NULL) {
f99161 ▷   ▷   ▷   nastyHash[i] = cur->next;
782515 ▷   ▷   ▷   cur->next = head;
a3d1de ▷   ▷   ▷   head = cur;
1ØØfe4 ▷   ▷   }
419371 ▷   }
dd6cfa ▷   /* Recompute CRCs for the list and redistribute them among the buckets */
d589b4 ▷   while (head) {
d2a8e3 ▷   ▷   cur = head;
ac1Ø85 ▷   ▷   head = head->next;
Øbeae1 ▷   ▷   len = strlen(cur->line);
Ø673e3 ▷   ▷   crc = CalculateCRC(poly, Ø, (byte const *)cur->line, len);
79e563 ▷   ▷   crc = AdvanceCRC(poly, crc, '\n');
26a9e1 ▷   ▷   cur->crc = crc;
4733e4 ▷   ▷   cur->next = nastyHash[crc % NASTY␣HASH␣SIZE];
3a523e ▷   ▷   nastyHash[crc % NASTY␣HASH␣SIZE] = cur;
3c9371 ▷   }
9a91de ▷   nastyPoly = poly;
c8efe6 }
34af5a
f7d837 /* Read in the nastylines file */
242978 static void
1fdbda ReadNasties(FILE *f)
41bb36 {
57658f ▷   char buf[128];
11af5a
82d99d ▷   while (fgets(buf, sizeof(buf)-1, f))
5ceØ21 ▷   ▷   AddNasty(buf);
46efe6 }
a7af5a
5Ø38e5 /*
9bff78 ·* Convert an encoded string to binary.
bab32c ·* No error checking is performed.
7e495d ·*/
86de46 static word32
c37467 GetWord32(EncodeFormat const *format, char const *buf, int len)
25bb36 {
691c93 ▷   word32 w = Ø;
Ø2af5a
59e87e ▷   while (len--)
435d3a ▷   ▷   w = (w<<format->bitsPerDigit) + DecodeDigit(format, *buf++);
d8b418 ▷   return w;
4defe6 }
b4af5a
```

```
aa9135 /* Attempt nasty line substitutions */
952978 static void
f2f2ae TryNasty(struct ParseNode *parent, Heap *heap, char const *limit)
Ø7bb36 {
b96d8f ▷   struct NastyLine const *nasty;
3c19ba ▷   struct Substitution const *subst;
89c266 ▷   struct ParseNode *child;
67de5f ▷   char const *end;
c46c3f ▷   EncodeFormat const *format = pnFormat(parent);
178bb8 ▷   OutputHandle oh;
f7cac8 ▷   char buf[4];
a44f88 ▷   CRC check;
f14ba4 ▷   int i;
1daf5a
57c47a ▷   /* Make sure the lines are hashed properly */
Ø67b52 ▷   if (nastyPoly != format->lineCRC)
Ø11d68 ▷   ▷   RehashNasties(format->lineCRC);
b2af5a
Øcd7f7 ▷   /* Get the line to be replaced */
e5f2f9 ▷   assert(parent->ps.pos == PREFIX_LENGTH);
22e465 ▷   end = memchr(parent->input, '\n', limit - parent->input);
57a6Øf ▷   if (!end)
aØ3a97 ▷   ▷   end = limit;
622d4e ▷   /* Get the line's check value */
a2Ø6Ø7 ▷   OutputInit(&oh, parent, NULL);
Ø6efba ▷   (void)OutputGetPrev(&oh);
8Ø9cØd ▷   i = 4;
Ø6d27Ø ▷   while (--i)
fbe9cd ▷   ▷   buf[i] = OutputGetPrev(&oh);
876428 ▷   check = GetWord32(format, buf, 4);
17ea4f ▷   /* Find the matches */
c562f5 ▷   nasty = nastyHash[check % NASTY_HASH_SIZE];
41245c ▷   for (; nasty; nasty = nasty->next) {
a67Ø7a ▷   ▷   if (nasty->crc == check) {
ac45c4 ▷   ▷   ▷   subst = SubstNasty(parent->input, end-parent->input,
b91269 ▷   ▷   ▷   ▷   ▷   ▷   ▷   ···nasty->line, Ø);
376cb9 ▷   ▷   ▷   if (subst) {
18b674 ▷   ▷   ▷   ▷   child = AddChild(parent, subst, NASTY_COST);
8efb2e ▷   ▷   ▷   ▷   if (child) {
533d9c ▷   ▷   ▷   ▷   ▷   child->ps.flags ¦= PS_FLAG_DYNAMIC;
1Øc6d6 ▷   ▷   ▷   ▷   ▷   HeapInsert(heap, &child->cost);
a12613 ▷   ▷   ▷   ▷   }
32a3a6 ▷   ▷   ▷   }
1dØfe4 ▷   ▷   }
b59371 ▷   }
22efe6 }
2Øaf5a
cc38e5 /*
c4e7cf ·* Form all of a ParseNode's children and add them to the heap.
a38c9Ø ·* Limit is the limit of allowable lookahead.
2e495d ·*/
562978 static void
d41774 AddChildren(ParseNode *parent, Heap *heap, char const *limit)
Ø3bb36 {
e41c2f ▷   char c = parent->input[Ø];
7cdc58 ▷   Substitution *subst = substitutions[Ø][c & 255];
589c24 ▷   ParseNode *child;
ce7e1f ▷   HeapCost cost;
1daf5a
efØ7e7 /* If you want to make pure insertion substitutions, do that here */
1baf5a
f4fdeb ▷   assert(parent->input < limit);▷ /* We always have at least one char */
86af5a
Øb147e ▷   while (subst) {
8e4e68 ▷   ▷   if (subst->inlen == 1 ¦¦▷   /* Easy case */
22e7Ø5 ▷   ▷   ▷   ((size_t)(limit-parent->input) >= subst->inlen &&
74Ø5b5 ▷   ▷   ▷   ·memcmp(subst->input, parent->input, subst->inlen) == Ø))
fc5b34 ▷   ▷   {
4dc51a ▷   ▷   ▷   cost = subst->cost;
Øc5f15 ▷   ▷   ▷   if (subst->filter)
2652f5 ▷   ▷   ▷   ▷   cost = subst->filter(parent, limit, subst);
bØ3Øa6 ▷   ▷   ▷   child = AddChild(parent, subst, cost);
```

```
186e75 ▷   ▷   ▷   if (child)
1f5097 ▷   ▷   ▷   ▷   HeapInsert(heap, &child->cost);
9c0fe4 ▷   ▷   }
31bb6f ▷   ▷   subst = subst->next;
8e9371 ▷   }
a3af5a
b707b2 ▷   /* Whole-line substitutions */
ec5e86 ▷   if (parent->ps.pos == PREFIX_LENGTH)
aca01d ▷   ▷   TryNasty(parent, heap, limit);
d5efe6 }
6baf5a
a6af5a
207ba4 /* cost if this line has a dynamic substitution, otherwise cost2 */
e68329 HeapCost
b353c1 FilterIsDynamic(struct ParseNode *parent, char const *limit,
3dbf9b ▷   struct Substitution const *subst)
bebb36 {
e4e8a7 ▷   (void)limit;
7b00fc ▷   return (parent->ps.flags & PS_FLAG_DYNAMIC) ? subst->cost : subst->cost2;
b5efe6 }
98af5a
7c7e51 /* cost if the current page is binary mode, else cost2 */
438329 HeapCost
02193e FilterIsBinary(struct ParseNode *parent, char const *limit,
69bf9b ▷   struct Substitution const *subst)
91bb36 {
8705 8c ▷   (void)parent; (void)limit;
16cfa1 ▷   return perpage.tabsize ? subst->cost2 : subst->cost;
26efe6 }
74af5a
34cdc0 /* Debugging utility */
f0a5c4 #define DEBUG 1▷/* Set to 1 to print every line considered */
24af5a
3e92b2 static size_t lastlen = 0;
72af5a
8e2978 static void
4e91b1 OverstrikeLine(char const *line, size_t len)
1abb36 {
65869a ▷   static size_t lastlen = 0;
917d22 ▷   int blanklen;
eaaf5a
c2577c ▷   if (!line) {
985676 ▷   ▷   if (lastlen)
7290 6d ▷   ▷   ▷   putchar('\n');
6037 12 ▷   ▷   lastlen = 0;
2ff7fd ▷   } else if (len ¦¦ lastlen) {
2775be ▷   ▷   if (len > 79)
2cba76 ▷   ▷   ▷   len = 79;
ff6dca ▷   ▷   blanklen = (lastlen > len) ? (int)lastlen - len : 0;
fba46c ▷   ▷   printf("%.*s%*s\r", (int)len, line, blanklen, "");
4411e2 ▷   ▷   fflush(stdout);
6a6451 ▷   ▷   lastlen = len;
0a9371 ▷   }
f9efe6 }
56af5a
c01a3f /* Print everything, for debugging */
ea2978 static void
f08788 PrintLine(char const *line, size_t len)
c4bb36 {
7ced71 ▷   if (line) {
f4f179 ▷   ▷   printf("%.*s\n", (int)len, line);
ad3112 ▷   ▷   lastlen = 0;
039371 ▷   }
51efe6 }
d1af5a
a4ef62 static HeapCost ParseAdvanceString(Heap *heap, ParseNode *pn);
11af5a
3138e5 /*
7cf89e ▷* Copy the parsechain from tail up to root, and hang it off of
3ef39a ▷* newroot, adjusting the costs and parse state accordingly. ▷Returns
7ebe79 ▷* NULL if it is unable to (invalid parse, too expensive, etc.)
bf85d0 ▷* Note that as per the convention, ParseAdvanceString is *not* called
```

```
9ed199 ‘* on the new tail node (but is called on all its parents).
6f495d ‘*/
f443e9 static ParseNode *
699e42 CopyParse(ParseNode const *tail, ParseNode const *root, ParseNode *newroot)
5fbb36 {
e7ac81 ▷   ParseNode *newtail, *parent;
11af5a
8939d6 ▷   if (tail == root)
9ca54Ø ▷   ▷   return newroot;
5a71d9 ▷   parent = CopyParse(tail->parent, root, newroot);
f6db9d ▷   if (!parent)
a87ed2 ▷   ▷   return NULL;
Ød978b ▷   newtail = AddChild(parent, tail->subst, ParseAdvanceString(NULL, parent));
77Ø3b5 ▷   NodeRelease(parent);
8eae3b ▷   return newtail;
82efe6 }
aØaf5a
d338e5 /*
7bba65 ‘* Replace oldnode with a dynamic substitution for newchar, if possible,
76cfdØ ‘* and fill in the chain down to "tail" just like before, but with no branches.
3d12Øf ‘* Add the resultant ParseNode to the heap.
9f495d ‘*/
982978 static void
d2a7ce AddDynamic(Heap *heap, ParseNode const *oldnode, ParseNode const *tail,
8255d4 ▷   int newchar)
83bb36 {
d222aa ▷   Substitution const *subst = oldnode->subst;
26Ø4de ▷   ParseNode *newnode;
f9af5a
7e3221 ▷   /* Only replace one-character substitutions */
a2bed9 ▷   if (subst->outlen != 1)
1299bf ▷   ▷   return;
bdaf5a
689552 ▷   subst = SubstDynamic(oldnode->subst->input, SubstString(newchar), Ø);
5a7b1a ▷   newnode = AddChild(oldnode->parent, subst, -1); /* Try it immediately */
1fd16Ø ▷   if (newnode) {
ffe1bd ▷   ▷   newnode->ps.flags ¦= PS_FLAG_DYNAMIC;
175b8Ø ▷   ▷   newnode = CopyParse(tail, oldnode, newnode);
c2f74b ▷   ▷   if (newnode)
7Ø5138 ▷   ▷   ▷   HeapInsert(heap, &newnode->cost);
879371 ▷   }
25efe6 }
67af5a
fb38e5 /*
953eeb ‘* Do the same, at a given (1-based) position on the line. ‘Owing to
892Øbe ‘* a minor glitch, we must never count the tail node, as this has not
489c71 ‘* been parsed yet, so its oldnode->ps.pos field is inaccurate.
71495d ‘*/
832978 static void
d5c9eØ AddDynamicAt(Heap *heap, int position, ParseNode const *tail, int newchar)
4cbb36 {
eØ8673 ▷   ParseNode const *oldnode = tail;
b3af5a
b243ab ▷   do {
ab6432 ▷   ▷   oldnode = oldnode->parent;
16f2ca ▷   } while (oldnode->ps.pos > position);
59af5a
de67f6 ▷   if (oldnode->ps.pos == position)
8dØ6d2 ▷   ▷   AddDynamic(heap, oldnode, tail, newchar);
35efe6 }
29af5a
Ød38e5 /*
Ø8b4Ø8 ‘* Given the computed and input check fields, correct the header field
455cf6 ‘* that *ends* at the given pos. ‘This can be used for both the line and
dd2b3d ‘* page CRC errors by jyst changing the pos. ‘(It relies on the fact
569ae5 ‘* that the page CRC fragment fits into the LineCRC type.)
bef1a8 ‘* It also relies on the fact that the CRC is at most 4 digits.
ac495d ‘*/
b72978 static void
594f87 ErrorCorrectHeader(Heap *heap, ParseNode const *tail, int pos,
58dbcØ ▷   EncodeFormat const *format, CRC crc, CRC check)
37bb36 {
```

```
974b6Ø ▷   CRC syndrome = crc ^ check;
bfaf5a
a2ae1a ▷   /* Find the position and the crc digit at that position */
282de1 ▷   while (syndrome >= (CRC)format->radix) {
69dcc3 ▷   ▷   if (syndrome & (CRC)(format->radix - 1))
eØ3fØa ▷   ▷   ▷   return;▷/* uncorrectable */
ccØ481 ▷   ▷   pos--;
867463 ▷   ▷   crc >>= format->bitsPerDigit;
846bca ▷   ▷   syndrome >>= format->bitsPerDigit;
8d9371 ▷   }
84e8Øa ▷   /* Paste in the correct digit */
cf785b ▷   AddDynamicAt(heap, pos, tail, EncodeDigit(format, crc & (format->radix-1)));
49efe6 }
f7af5a
8338e5 /*
fb4732 ‣* This function walks back through the line, and if the line CRC could be
eee94b ‣* made correct by changing a character to another legal character,
9afc4b ‣* the change is added (on probation) to the substitution table.
b9495d ‣*/
fØ2978 static void
b7Ød7f ErrorCorrect(Heap *heap, OutputHandle oh, EncodeFormat const *format,
fd993a ▷   CRC syndrome)
63bb36 {
ØdbØ2f ▷   ParseNode const *tail = oh.node;
e638de ▷   int c;
4eaf5a
5f8d95 ▷   syndrome = ReverseCRC(format->lineCRC, syndrome, Ø);
3af986 ▷   while (oh.node->ps.pos > PREFIX_LENGTH) {
e66468 ▷   ▷   c = OutputGetPrev(&oh);
474da2 ▷   ▷   if (c == '\n' ¦¦ c == -1) {▷/* Can't happen */
Øaf159 ▷   ▷   ▷   printf("Line ended at pos %d\n", oh.node->ps.pos);
deØecc ▷   ▷   ▷   return;
ddØfe4 ▷   ▷   }
6cab2Ø ▷   ▷   syndrome = ReverseCRC(format->lineCRC, syndrome, Ø);
1cb749 ▷   ▷   if (syndrome >= Øx1ØØ ¦¦ !substitutions[Ø][c^syndrome] ¦¦
Øaaed6 ▷   ▷   ▷   oh.node->subst->outlen != 1)
67dØ4Ø ▷   ▷   ▷   ▷   continue;
443eaf ▷   ▷   AddDynamic(heap, oh.node, tail, c^syndrome);
389371 ▷   }
83efe6 }
88af5a
5Ø38e5 /*
68ecf3 ‣* Parsing operations. ‣This is a rather ugly and ad-hoc parser that
e713ee ‣* knows a lot about the fixed-field format produced by the munge
Øa7eb9 ‣* utility. ‣The main state variable is the position in
653Øbc ‣* the line, which controls the expected header, the position of
bcfdee ‣* tab stops, and the maximum permissible line length.
a4495d ‣*/
b63bf3 #define OCCASIONALLY 1ØØ
53af5a
f627Ø4 /* Set up a ParseState to top-of-page */
fb2978 static void
182c82 ParseStateInit(ParseState *ps)
cdbb36 {
a42921 ▷   static struct ParseState const parseNull = { Ø, Ø, Ø };
2d8879 ▷   *ps = parseNull;
bbefe6 }
22af5a
fd38e5 /*
794d91 ‣* Try to accept a newline, checking CRCs and even doing error-correction
8Ø84Ø9 ‣* as appropriate.
f3495d ‣*/
c2beb7 static int
d73dØb ParseNewline(Heap *heap, ParseNode *pn, char const *string)
c5bb36 {
1c8bb8 ▷   OutputHandle oh;
9e38de ▷   int c;
bØ4df1 ▷   char debugbuf[PREFIX_LENGTH+LINE_LENGTH+1Ø];
c41bØ1 ▷   char *header, *body, *end;
5c33d5 ▷   int pos, width;
eØ8143 ▷   CRC crc, check;
45ede8 ▷   ParseNode *temp;
```

```
7fd93a ▷   static int occasionally = OCCASIONALLY;
394ab4 ▷   EncodeFormat const *format = pnFormat(pn);
95fa06 ▷   EncodeFormat const *headerFormat = &hexFormat;
ecaf5a
18439a ▷   /* Get the line into a buffer for analysis */
d8c9fe ▷   OutputInit(&oh, pn, string);
71d40a ▷   end = debugbuf + sizeof(debugbuf)-1;
eba54a ▷   header = OutputGetUntil(oh, end, '\n');
908947 ▷   /* Strip leading and trailing whitespace */
52b66e ▷   while (header < end && isspace((unsigned char)header[0]))
b94fd9 ▷   ▷   header++;
447bb8 ▷   while (header < end && isspace((unsigned char)end[-1]))
b4c15c ▷   ▷   end--;
5996ac ▷   *end++ = '\n';
69af5a
1deb67 ▷   /* Start of checksummed area */
a4a599 ▷   body = header + PREFIX_LENGTH;
dbd996 ▷   /* Blank lines are missing the trainign space from the prefix */
b2ed6d ▷   if (body >= end)
1cd5f4 ▷   ▷   body = end-1;
45af5a
2094d4 ▷   crc = CalculateCRC(format->lineCRC, 0, body, end-body);
ad0e63 ▷   check = GetWord32(format, header+2, 4);
5aa6f0 ▷   if (crc != check) {
1470e ▷   ▷   if (!--occasionally) {
3097cf ▷   ▷   ▷   OverstrikeLine(header, end-header-1);
1c58ef ▷   ▷   ▷   occasionally = OCCASIONALLY;
ba0fe4 ▷   ▷   }
8680e ▷   ▷   /* Try ECC on the line */
32b3cd ▷   ▷   /* If we haven't already tried ECC on the line... */
51c23c ▷   ▷   if (!(pn->ps.flags & PS_FLAG_DYNAMIC)) {
b15a63 ▷   ▷   ▷   ErrorCorrectHeader(heap, pn, PREFIX_LENGTH-1, format, crc, check);
d81377 ▷   ▷   ▷   ErrorCorrect(heap, oh, format, crc ^ check);
a00fe4 ▷   ▷   }
d10283 ▷   ▷   return COST_INFINITY;
1e9371 ▷   }
7070bd ▷   /* Good enough that we always print it */
96e134 ▷   OverstrikeLine(header, end-header-1);
e3af5a
24a318 ▷   /* Okay, now there are two cases - header line or running CRC */
a7578a ▷   if (pn->ps.flags & PS_FLAG_INHEADER) {
0cf8ba ▷   ▷   /* Do things for first header */
1211c2 ▷   ▷   if (!(pn->ps.flags & PS_FLAG_PASTHEADER)) {
d24811 ▷   ▷   ▷   /* Check version number */
b752a4 ▷   ▷   ▷   width = EncodedLength(headerFormat, HDR_VERSION_BITS);
8a9328 ▷   ▷   ▷   c = (int)GetWord32(&hexFormat, body, width);
319f14 ▷   ▷   ▷   if (c != 0) {
92c1d5 ▷   ▷   ▷   ▷   fputs("Fatal: you need a newer version of repair"
fcf527 ▷   ▷   ▷   ▷   ^^^^^^" to process this file\n", stderr);
55bbe9 ▷   ▷   ▷   ▷   exit(1);
1fa3a6 ▷   ▷   ▷   }
4c0367 ▷   ▷   ▷   /* Suck in page CRC, after version & flags */
646f44 ▷   ▷   ▷   pos = width + EncodedLength(headerFormat, HDR_FLAG_BITS);
6830d ▷   ▷   ▷   width = EncodedLength(headerFormat, format->pageCRC->bits);
7602d5 ▷   ▷   ▷   perpage.page_check = GetWord32(&hexFormat, body+pos, width);
43419f ▷   ▷   ▷   /* Get tab size */
054b12 ▷   ▷   ▷   pos += width;
dbcebf ▷   ▷   ▷   width = EncodedLength(headerFormat, HDR_TABWIDTH_BITS);
e152ad ▷   ▷   ▷   perpage.tabsize = GetWord32(&hexFormat, body+pos, width);
d2af5a
f908be ▷   ▷   ▷   /* Once we have the header, don't reconsider */
d467fb ▷   ▷   ▷   if (!(pn->ps.flags & PS_FLAG_PASTHEADER))
b15692 ▷   ▷   ▷   ▷   while ((temp = (ParseNode *)HeapGetMin(heap)) != NULL)
081882 ▷   ▷   ▷   ▷   ▷   NodeRelease(temp);
2ef4cd ▷   ▷   ▷   pn->ps.page_crc = 0;▷   /* Clear for top of page */
d80fe4 ▷   ▷   }
87f037 ▷   } else {
b5f960 ▷   ▷   /* Check the CRC-32 */
3a845e ▷   ▷   crc = CalculateCRC(format->pageCRC, pn->ps.page_crc, body, end-body);
9687eb ▷   ▷   pn->ps.page_crc = crc;
33bcaf ▷   ▷   crc = RunningCRCFromPageCRC(format, crc);
bb9b68 ▷   ▷   check = GetWord32(format, header, 2);
```

```
de5Ø95 ▷   ▷   if (crc != check) {
4Ø5981 ▷   ▷   ▷   if (!(pn->ps.flags & PS_FLAG_DYNAMIC))
f3dc3d ▷   ▷   ▷   ▷   ErrorCorrectHeader(heap, pn, 2, format, crc, check);
Ø7358b ▷   ▷   ▷   return COST_INFINITY;
9eØfe4 ▷   ▷   }
9c9371 ▷   }
4baf5a
8d34Ø6 ▷   /* Hey, it's correct! */
9Ø8f23 ▷   PrintLine(header, end-header-1);
b7af5a
8da68Ø ▷   /* Start next line */
8ec47d ▷   pn->ps.pos = Ø;
ff4c54 ▷   /* Clear most other flags, but we *have* got a header */
511321 ▷   c = pn->ps.flags & PS_FLAG_DYNAMIC;
aØ555Ø ▷   pn->ps.flags &= PS_FLAG_BINEND ¦ PS_MASK_FORMAT;
32226e ▷   pn->ps.flags ¦= PS_FLAG_PASTHEADER;
8ed8af ▷   /*
e5a7cd ▷   ·* Give a bonus to the next line for having completed this one,
5247ea ▷   ·* less if it was dynamically fixed.
b3a4e3 ▷   ·*/
a3125d ▷   return c ? COST_LINE : COST_LINE*2/3;
47efe6 }
32af5a
e438e5 /*
33d1a6 ·* Advance the parse state with pointed-to character. ·Returns
b7a7Øf ·* COST_INFINITY if an impossible state is reached, otherwise returns a
d85d1e ·* cost value. ·(Normally Ø, this can be increased to penalize unlikely
22af6d ·* output combinations to nudge the correction in a certain direction.)
11495d ·*/
f5b5fa static HeapCost
Øfbd15 ParseAdvance(Heap *heap, ParseNode *pn, char const *string)
Øebb36 {
843e38 ▷   int i, retval = Ø;
9Ød6ed ▷   char c = *string;
514ab4 ▷   EncodeFormat const *format = pnFormat(pn);
1daf5a
d8d8af ▷   /*
a73ceØ ▷   ·* Insist on spaces being correctly converted to SPACE_CHAR.
Ø7eØ1c ▷   ·* There's a little irregularity just before EOL.
3e1a3d ▷   ·* Line contiunation and formfeed are also only legal at EOL.
12a4e3 ▷   ·*/
4b599Ø ▷   if (c == ' ') {
1Ø7a25 ▷   ▷   if (pn->ps.flags & PS_FLAG_SPACE && !(pn->ps.flags & PS_FLAG_TAB))
e22f83 ▷   ▷   ▷   pn->ps.flags ¦= PS_FLAG_EOL;
579128 ▷   ▷   pn->ps.flags ¦= PS_FLAG_SPACE;
61b68Ø ▷   } else if (pn->ps.flags & PS_FLAG_EOL) {
b8e1a9 ▷   ▷   if (c != '\n')
d6358b ▷   ▷   ▷   return COST_INFINITY;
be98b9 ▷   } else if (c == SPACE_CHAR) {
588Ø6d ▷   ▷   if (!(pn->ps.flags & PS_FLAG_SPACE))
722f83 ▷   ▷   ▷   pn->ps.flags ¦= PS_FLAG_EOL;
a38a86 ▷   } else if (c == CONTIN_CHAR ¦¦ c == FORMFEED_CHAR) {
ec2f83 ▷   ▷   ▷   pn->ps.flags ¦= PS_FLAG_EOL;
adfØ37 ▷   } else {
5ae39c ▷   ▷   pn->ps.flags &= ~PS_FLAG_SPACE;
999371 ▷   }
c3af5a
f4Ø9b1 ▷   switch (pn->ps.pos) {
a4e2df ▷   ▷   case Ø:
7793f1 ▷   ▷   ▷   if (c == ' ' ¦¦ c == '\n') {
ff395e ▷   ▷   ▷   ▷   break;▷ ▷   /* Ignore ws and blank lines completely */
b8676d ▷   ▷   ▷   } else if (c == '\f' ¦¦ c == HDR_PREFIX_CHAR) {
3edd6Ø ▷   ▷   ▷   ▷   /* Start of a new page */
e68111 ▷   ▷   ▷   ▷   pn->ps.flags ¦= PS_FLAG_INHEADER;▷  /* Expect header next */
a24dØ3 ▷   ▷   ▷   ▷   if (c == '\f')
fe14c3 ▷   ▷   ▷   ▷   ▷   break;
1Øe11a ▷   ▷   ▷   ▷   /* And fall through to increment pos */
d82b63 ▷   ▷   ▷   } else if (pn->ps.flags & PS_FLAG_INHEADER ¦¦
f5f74c ▷   ▷   ▷   ▷   ▷   ···pn->ps.flags & PS_FLAG_BINEND ¦¦
45c869 ▷   ▷   ▷   ▷   ▷   ···!(pn->ps.flags & PS_FLAG_PASTHEADER) ¦¦
9b44bc ▷   ▷   ▷   ▷   ▷   ···DecodeDigit(format, c) < Ø) {
76b6cf ▷   ▷   ▷   ▷   return COST_INFINITY;▷  /* Various illegal cases */
```

```
cfa3a6 ▷   ▷   ▷   }
c27fef ▷   ▷   ▷   pn->ps.pos++;
edcdcb ▷   ▷   ▷   break;
53b8Ø3 ▷   ▷   case 1:
d584f7 ▷   ▷   ▷   if ((pn->ps.flags & PS_FLAG_INHEADER)) {
7e44c6 ▷   ▷   ▷   ▷   format = FindFormat(c);▷/* Second char of header */
8837ab ▷   ▷   ▷   ▷   if (!format)
e7e616 ▷   ▷   ▷   ▷   ▷   return COST_INFINITY;
cee771 ▷   ▷   ▷   ▷   i = registerFormat(format);
6a8af3 ▷   ▷   ▷   ▷   psSetFormat(&pn->ps, i);
86f6bb ▷   ▷   ▷   ▷   pn->ps.pos++;
cØf622 ▷   ▷   ▷   ▷   break;
78a3a6 ▷   ▷   ▷   }
Øed4eØ ▷   ▷   ▷   if (DecodeDigit(format, c) < Ø)
c22eØ9 ▷   ▷   ▷   ▷   return COST_INFINITY;▷  /* Illegal */
eb7fef ▷   ▷   ▷   pn->ps.pos++;
1dcdcb ▷   ▷   ▷   break;
dd5767 ▷   ▷   case 2:
73Ødbb ▷   ▷   case 3:
6681be ▷   ▷   case 4:
ØØe217 #if PREFIX_LENGTH != 7
e79691 #error fix this code
277454 #endif
6cad19 ▷   ▷   case PREFIX_LENGTH-2:
3Ød4eØ ▷   ▷   ▷   if (DecodeDigit(format, c) < Ø)
c42eØ9 ▷   ▷   ▷   ▷   return COST_INFINITY;▷  /* Illegal */
a67fef ▷   ▷   ▷   pn->ps.pos++;
3acdcb ▷   ▷   ▷   break;
25427d ▷   ▷   case PREFIX_LENGTH-1:
Ø4d5Ø1 ▷   ▷   ▷   if (c == ' ') {
b2f6bb ▷   ▷   ▷   ▷   pn->ps.pos++;
7bf622 ▷   ▷   ▷   ▷   break;
f6f4f7 ▷   ▷   ▷   } else if (c != '\n') {
44daf8 ▷   ▷   ▷   ▷   return COST_INFINITY;
34a3a6 ▷   ▷   ▷   }
c5a7eØ ▷   ▷   ▷   /* Blank lines may be missing this space char */
b8Øafa ▷   ▷   ▷   /*FALLTHROUGH*/
c8c7ed ▷   ▷   /* The normal line starts here, at position 7 */
579ae3 ▷   ▷   default:
2Ø65ff ▷   ▷   ▷   if (pn->ps.flags & PS_FLAG_INHEADER) {▷ /* Header line */
821Øa2 ▷   ▷   ▷   ▷   /* Format is "--abcd Ø123456789abcdefØ12 Page %u of %s" */
ØfbØd9 ▷   ▷   ▷   ▷   int off = pn->ps.pos - (PREFIX_LENGTH+HDR_ENC_LENGTH);
2bcee3 ▷   ▷   ▷   ▷   /* Offset relative to end of hex header */
8541c7 ▷   ▷   ▷   ▷   if (off < Ø) {
84f9cØ ▷   ▷   ▷   ▷   ▷   if (HexDigitValue(c & 255) < Ø)
3bdfd3 ▷   ▷   ▷   ▷   ▷   ▷   return COST_INFINITY;
97db3c ▷   ▷   ▷   ▷   } else if (off < 6) {
583845 ▷   ▷   ▷   ▷   ▷   if (c != " Page "[off])▷/* Yes, this is legal C */
cØdfd3 ▷   ▷   ▷   ▷   ▷   ▷   return COST_INFINITY;
fb7Ø9b ▷   ▷   ▷   ▷   } else if (off == 6) {
4845c5 ▷   ▷   ▷   ▷   ▷   if (c < '1' ¦¦ c > '9')▷/* First digit of page no. */
afdfd3 ▷   ▷   ▷   ▷   ▷   ▷   return COST_INFINITY;
3Øb5fc ▷   ▷   ▷   ▷   } else {
cb9a56 ▷   ▷   ▷   ▷   ▷   /* Re-base to end of scanned part of page number */
3cbØeØ ▷   ▷   ▷   ▷   ▷   off -= 7 + PageNumDigits(pn);
6Ø91c3 ▷   ▷   ▷   ▷   ▷   if (off == Ø) {
731d4e ▷   ▷   ▷   ▷   ▷   ▷   if (c >= 'Ø' && c <= '9' && PageNumDigits(pn) < 3)
6247Ø3 ▷   ▷   ▷   ▷   ▷   ▷   ▷   PageNumDigitsIncrement(pn);
ac2945 ▷   ▷   ▷   ▷   ▷   ▷   else if (c != ' ')
fe9Øed ▷   ▷   ▷   ▷   ▷   ▷   ▷   return COST_INFINITY;
aØf15a ▷   ▷   ▷   ▷   ▷   } else if (off < 4) {
23da3f ▷   ▷   ▷   ▷   ▷   ▷   if (c != " of "[off])
5a9Øed ▷   ▷   ▷   ▷   ▷   ▷   ▷   return COST_INFINITY;
59685f ▷   ▷   ▷   ▷   ▷   } else if (off == 4) {
81Øb75 ▷   ▷   ▷   ▷   ▷   ▷   if (!isgraph(c))
749Øed ▷   ▷   ▷   ▷   ▷   ▷   ▷   return COST_INFINITY;
cbbc33 ▷   ▷   ▷   ▷   ▷   } else if (c < ' ' ¦¦ (c & 255) > '~') {
b8daec ▷   ▷   ▷   ▷   ▷   ▷   if (c != '\n')
449Øed ▷   ▷   ▷   ▷   ▷   ▷   ▷   return COST_INFINITY;
1771a6 ▷   ▷   ▷   ▷   ▷   ▷   return ParseNewline(heap, pn, string);
221c89 ▷   ▷   ▷   ▷   ▷   }
3a2613 ▷   ▷   ▷   ▷   }
```

```
c2f966 ▷   ▷   ▷   } else if (!perpage.tabsize) {▷ /* Radix-64 line */
47f3Øb ▷   ▷   ▷   ▷   /* Line is "RlNFVF9UQU== ‘‘\n" */
964fe3 ▷   ▷   ▷   ▷   if (isspace(c & 255)) {
81566d ▷   ▷   ▷   ▷   ▷   if (!(pn->ps.flags & PS_FLAG_BINWS)) {
b8a474 ▷   ▷   ▷   ▷   ▷   ▷   if ((pn->ps.pos - PREFIX_LENGTH) % 4 != Ø)
3e9Øed ▷   ▷   ▷   ▷   ▷   ▷   ▷   return COST_INFINITY;
41Ø24b ▷   ▷   ▷   ▷   ▷   ▷   pn->ps.flags ¦= PS_FLAG_BINWS;
7214fd ▷   ▷   ▷   ▷   ▷   ▷   if (pn->ps.pos - PREFIX_LENGTH < BYTES_PER_LINE*4/3)
Øa535b ▷   ▷   ▷   ▷   ▷   ▷   ▷   pn->ps.flags ¦= PS_FLAG_BINEND;
6e1c89 ▷   ▷   ▷   ▷   ▷   }
5a24a4 ▷   ▷   ▷   ▷   ▷   if (c == ’\n’)
b271a6 ▷   ▷   ▷   ▷   ▷   ▷   return ParseNewline(heap, pn, string);
7Ø82bc ▷   ▷   ▷   ▷   } else if (pn->ps.flags & PS_FLAG_BINWS) {
aee616 ▷   ▷   ▷   ▷   ▷   return COST_INFINITY;
26f1b7 ▷   ▷   ▷   ▷   } else if (c == RADIX64_END_CHAR) {
dc2b85 ▷   ▷   ▷   ▷   ▷   if ((pn->ps.pos - PREFIX_LENGTH) % 4 < 2)
6ddfd3 ▷   ▷   ▷   ▷   ▷   ▷   return COST_INFINITY;
fØ89aØ ▷   ▷   ▷   ▷   ▷   pn->ps.flags ¦= PS_FLAG_BINEND;
f442df ▷   ▷   ▷   ▷   } else if (pn->ps.flags & PS_FLAG_BINEND) {
c5e616 ▷   ▷   ▷   ▷   ▷   return COST_INFINITY;
Ø34dcc ▷   ▷   ▷   ▷   } else if (Radix64DigitValue(c) < Ø) {
f5e616 ▷   ▷   ▷   ▷   ▷   return COST_INFINITY;
2c2613 ▷   ▷   ▷   ▷   }
55ee67 ▷   ▷   ▷   } else {▷   /* Normal line */
c263c1 ▷   ▷   ▷   ▷   /* Make sure tab stops come out right */
eØ8639 ▷   ▷   ▷   ▷   if (pn->ps.flags & PS_FLAG_TAB) {
ec1899 ▷   ▷   ▷   ▷   ▷   if (((pn->ps.pos - PREFIX_LENGTH) % perpage.tabsize) == Ø)
567dbd ▷   ▷   ▷   ▷   ▷   ▷   pn->ps.flags &= ~PS_FLAG_TAB;
fe96fd ▷   ▷   ▷   ▷   ▷   else if (c != TAB_PAD_CHAR && c != ’\n’) {
9acf7e ▷   ▷   ▷   ▷   ▷   ▷   return COST_INFINITY;▷  /* Illegal */
d21c89 ▷   ▷   ▷   ▷   ▷   }
Øb2613 ▷   ▷   ▷   ▷   }
82980e ▷   ▷   ▷   ▷   /*
9ebfb5 ▷   ▷   ▷   ▷   ‘* Yes, this code has hard-coded ASCII assumptions
42ef16 ▷   ▷   ▷   ▷   ‘* It knows that the acceptable range of ’\n’, ’ ’..’~’,
f58Ød5 ▷   ▷   ▷   ▷   ‘* TAB_CHAR, FORMFEED_CHAR is in that order.
3616Øe ▷   ▷   ▷   ▷   ‘* Signed char machines have it backwards, to be confusing.
2f1Ø2Ø ▷   ▷   ▷   ▷   ‘*/
342566 ▷   ▷   ▷   ▷   if ((c & 255) < ’ ’) {
f74512 ▷   ▷   ▷   ▷   ▷   /* Newline! (Or something illegal) */
e79b1f ▷   ▷   ▷   ▷   ▷   if (c != ’\n’)
Øbdfd3 ▷   ▷   ▷   ▷   ▷   ▷   return COST_INFINITY;
867144 ▷   ▷   ▷   ▷   ▷   return ParseNewline(heap, pn, string);
752613 ▷   ▷   ▷   ▷   }
bfbee8 ▷   ▷   ▷   ▷   /* A normal character */
81c75f ▷   ▷   ▷   ▷   if ((c & 255) > ’~’) {
57db68 ▷   ▷   ▷   ▷   ▷   if (pn->ps.flags & PS_FLAG_INHEADER)
adcf7e ▷   ▷   ▷   ▷   ▷   ▷   return COST_INFINITY;▷  /* Illegal */
dØ4539 ▷   ▷   ▷   ▷   ▷   if (c == TAB_CHAR)
8729eØ ▷   ▷   ▷   ▷   ▷   ▷   pn->ps.flags ¦= PS_FLAG_TAB;
f25a27 ▷   ▷   ▷   ▷   ▷   else if (c != FORMFEED_CHAR && c != SPACE_CHAR &&
78272b ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ‘‘‘c != CONTIN_CHAR)
c5cf7e ▷   ▷   ▷   ▷   ▷   ▷   return COST_INFINITY;▷  /* Illegal */
7Ø2613 ▷   ▷   ▷   ▷   }
e9a3a6 ▷   ▷   ▷   }
ae Ø993 ▷   ▷   ▷   if (++pn->ps.pos > PREFIX_LENGTH + LINE_LENGTH)
6fdaf8 ▷   ▷   ▷   ▷   return COST_INFINITY;
6fcdcb ▷   ▷   ▷   break;
7b9371 ▷   }
2a94b6 ▷   return retval;
f1efe6 }
Øfaf5a
6d38e5 /*
f37e29 ‘* Run the parser over the string in a ParseNode (using repeated calls
27d956 ‘* to ParseAdvance). ‘Return the penalty cost, or COST_INFINITY if
aa3b99 ‘* it’s impossible
21495d ‘*/
93b5fa static HeapCost
a48f6Ø ParseAdvanceString(Heap *heap, ParseNode *pn)
f1bb36 {
Ø5bd75 ▷   HeapCost cost, total = Ø;
175ebb ▷   char const *string = pn->subst->output;
```

```
e0af5a
d2Ø883 ▷   while (*string) {
3519ae ▷   ▷   cost = ParseAdvance(heap, pn, string++);
585c7f ▷   ▷   if (cost == COST_INFINITY)
5459a2 ▷   ▷   ▷   return cost;
2Øb55d ▷   ▷   total += cost;
ØØ9371 ▷   }
df9b53 ▷   return total;
9defe6 }
cbaf5a
9d7Ø7c static unsigned int *globalStats = NULL;
Ø98cØ4 static unsigned globalSize = Ø;
bb4b81 static unsigned globalEdits = Ø;
63af5a
7638e5 /*
e3fØ3c ‘* This walks the list of substitutions, performing two tasks with
e6552c ‘* the statistics gathered.
5c775e ‘*
Øae59c ‘* First, although not essential, it prints any interesting changes
b931da ‘* (non-identity substitutions) made, and a count of the total number
aa4a8a ‘* of substitutions (including identity) as an approximate character count.
62775e ‘*
c196a3 ‘* Second, it does maintenance on dynamic (learned during program
72667c ‘* execution) substitutions. ‘It discards any substitutions that end
76198b ‘* up unused, and computes nice costs for the others, based on the
ed8c72 ‘* global (per-file) statistics.
ba775e ‘*
16e1a7 ‘* (This function is also called at the end to print the per-file stats,
e9d225 ‘* which does redundant weight adjustment, but it’s harmless.)
93495d ‘*/
f12978 static void
3Ø451f UseStats(unsigned int *stats, FILE *log)
9ebb36 {
218dc2 ▷   unsigned int i, j, n, changes = Ø;
afa458 ▷   unsigned long grand = Ø;
bcØØd6 ▷   Substitution *s, **sp;
51af5a
6cfa34 ▷   if (!stats)
8Ø99bf ▷   ▷   return;
77af5a
f859f8 ▷   /* Yes, this loop is permuted on purpose */
a7d7ce ▷   for (j = Ø; j < elemsof(*substitutions); j++) {
e7f246 ▷   ▷   for (i = Ø; i < elemsof(substitutions); i++) {
35be6Ø ▷   ▷   ▷   sp = &substitutions[i][j];
cØ34d2 ▷   ▷   ▷   while ((s = *sp) != Ø) {
7fe95a ▷   ▷   ▷   ▷   grand += n = stats[s->index];
a236Øf ▷   ▷   ▷   ▷   /* Retain or purge dynamic substitutions, depending. */
bcaacf ▷   ▷   ▷   ▷   if (SubstIsDynamic(s)) {
6755ff ▷   ▷   ▷   ▷   ▷   if (n) {
3bb83a ▷   ▷   ▷   ▷   ▷   ▷   SubstAdjust(s, n);
7bcaØ4 ▷   ▷   ▷   ▷   ▷   } else if (!globalStats[s->index]) {
d51c57 ▷   ▷   ▷   ▷   ▷   ▷   /* Forget unused dynamic substitutions */
43Øe43 ▷   ▷   ▷   ▷   ▷   ▷   *sp = s->next;
72c9ed ▷   ▷   ▷   ▷   ▷   ▷   if (SubstIsNasty(s))
d334dd ▷   ▷   ▷   ▷   ▷   ▷   ▷   free((char *)s->input);▷/* Dynamically allocated */
26cØ2f ▷   ▷   ▷   ▷   ▷   ▷   SubstFree(s);
c16e1c ▷   ▷   ▷   ▷   ▷   ▷   continue;
481c89 ▷   ▷   ▷   ▷   ▷   }
4e2613 ▷   ▷   ▷   ▷   }
9be89Ø ▷   ▷   ▷   ▷   sp = &s->next;
1c98Øe ▷   ▷   ▷   ▷   /*
87dd2Ø ▷   ▷   ▷   ▷   ‘* Print interesting substitutions. ‘Some boring substitutions,
e1aaØ9 ▷   ▷   ▷   ▷   ‘* flagged with an index value of zero, are not printed.
981Ø2Ø ▷   ▷   ▷   ▷   ‘*/
29d577 ▷   ▷   ▷   ▷   if (!s->index ¦¦ !n)
3d5914 ▷   ▷   ▷   ▷   ▷   continue;
Ø4ac88 ▷   ▷   ▷   ▷   changes += n;
3a7265 ▷   ▷   ▷   ▷   fprintf(log, "\t%2ux \"%.*s\"%*s-> \"%.*s\"%*s(cost ",
3b3952 ▷   ▷   ▷   ▷   ▷   ‘‘‘stats[s->index], (int)s->inlen, s->input,
a11a46 ▷   ▷   ▷   ▷   ▷   ‘‘‘s->inlen>3 ? Ø : 3-(int)s->inlen, "",
f97aØ4 ▷   ▷   ▷   ▷   ▷   ‘‘‘(int)s->outlen, s->output,
559951 ▷   ▷   ▷   ▷   ▷   ‘‘‘s->outlen>3 ? Ø : 3-(int)s->outlen, "");
```

```
274b67 ▷   ▷   ▷   ▷   fprintf(log, s->cost == COST_INFINITY ? "-" : "%d", s->cost);
4d8adf ▷   ▷   ▷   ▷   if (s->filter)
b4a39f ▷   ▷   ▷   ▷   ▷   fprintf(log, s->cost2 == COST_INFINITY ? "/-" : "/%d",
aaØ536 ▷   ▷   ▷   ▷   ▷   ▪▪▪▪▪▪▪▪s->cost2);
bfØa7e ▷   ▷   ▷   ▷   fputs(SubstIsDynamic(s) ? ") ** LEARNED **\n" : ")\n", log);
cba3a6 ▷   ▷   ▷   }
4eØfe4 ▷   ▷   }
8Ø9371 ▷   }
826efØ ▷   fprintf(log, "\tTotal: %u changes (out of %lu)\n", changes, grand);
34efe6 }
29af5a
1Ø2978 static void
6aabØ8 DoStats(ParseNode const *node, unsigned int page, FILE *log)
ØØbb36 {
73Ø2e6 ▷   unsigned int *stats;
165a4a ▷   unsigned int n;
ØØaf5a
Ø2cØeØ ▷   /* Enlarge global stats if needed */
92bad5 ▷   if (globalSize < substCount) {
2ce939 ▷   ▷   stats = realloc(globalStats, substCount * sizeof(*stats));
Øab1d2 ▷   ▷   if (!stats) ▪{
e635fa ▷   ▷   ▷   fputs("Fatal error: out of memory for stats!\n", stderr);
ØaØa38 ▷   ▷   ▷   exit(1);
e6Øfe4 ▷   ▷   }
9344ef ▷   ▷   for (n = globalSize; n < substCount; n++)
6325Ø7 ▷   ▷   ▷   stats[n] = Ø;
85c8ad ▷   ▷   globalStats = stats;
51f714 ▷   ▷   globalSize = substCount;
9a9371 ▷   }
4eaf5a
261558 ▷   /* Allocate per-page stats */
c594e8 ▷   stats = calloc(substCount, sizeof(*stats));
8c426Ø ▷   if (!stats) {
9Ø2Ød9 ▷   ▷   fputs("Fatal error: out of memory for stats!\n", stderr);
144bb3 ▷   ▷   exit(1);
ed9371 ▷   }
Ø59414 ▷   /* Cheat and assume that calloc() initializes unsigned ints to zero */
256291 ▷   while (node) {
59a922 ▷   ▷   stats[node->subst->index]++;
d4a268 ▷   ▷   node = node->parent;
489371 ▷   }
d1af5a
da172Ø ▷   /* Keep the global counts accurate */
ed3e28 ▷   for (n = Ø; n < substCount; n++)
fØØd6e ▷   ▷   globalStats[n] += stats[n];
1aaf5a
263d2c ▷   fprintf(log, "Page %u substitutions:\n", page);
212a6c ▷   UseStats(stats, log);
ecaf5a
f3Ø2e7 ▷   free(stats);
75efe6 }
82af5a
73d5fa /* Spit out a page of data (needs work). ▪Returns number of lines */
c965bf static unsigned
e6Ø223 PrintPage(OutputHandle oh, FILE *out)
89bb36 {
14efdf ▷   char pagebuf[PAGE_BUFFER_SIZE];
3199Øe ▷   char *p1;▷  /* Beginning of current line */
2b9ØØ2 ▷   char *p2;▷  /* End of current line (WS stripped) */
7f226a ▷   char *p3;▷  /* End of current line (newline) */
8da99a ▷   char *p4;▷  /* End of all output */
54f836 ▷   unsigned lines = Ø;
31af5a
ea75db ▷   p4 = pagebuf + sizeof(pagebuf);
a85dØ7 ▷   p1 = OutputGetUntil(oh, p4, -1);
7aaf5a
3555aØ ▷   /* Output the lines without leading & trailing whitespace */
Ød4c13 ▷   while (p1 < p4) {
b3d3ae ▷   ▷   /* Identify the line */
e66d8c ▷   ▷   p3 = memchr(p1, '\n', p4-p1);
1c9ce5 ▷   ▷   if (!p3)
f2ce72 ▷   ▷   ▷   p3 = p4;
```

```
16f889 ▷   ▷   /* Delete leading whitespacee */
7cbØ4f ▷   ▷   while (isspace((unsigned char)*p1) && p1 < p3)
78377d ▷   ▷   ▷   p1++;
31ee7b ▷   ▷   /* Delete trailing whitepace */
f78243 ▷   ▷   p2 = p3;
eØ13c9 ▷   ▷   while (isspace((unsigned char)p2[-1]) && p1 < p2)
34b7f2 ▷   ▷   ▷   p2--;
a3deed ▷   ▷   /* Spit out this line */
a929c1 ▷   ▷   fwrite(p1, 1, (size_t)(p2-p1), out);
2aØ93f ▷   ▷   putc('\n', out);
6bbfb6 ▷   ▷   /* Advance p1 past the newline */
2Ø7a7f1 ▷   ▷   p1 = p3 + 1;
b6baf9 ▷   ▷   lines++;
719371 ▷   }
a9adda ▷   return lines;
43efe6 }
97af5a
688e61 static volatile int interrupt = Ø;
266cee static void (* volatile oldhandler)(int) = SIG_DFL;
ØØaf5a
a51bdd static void inthandler(int sig)
6fbb36 {
328e98 ▷   if (++interrupt > 2)
1b755b ▷   ▷   (void)signal(sig, oldhandler);
Ødefe6 }
c3af5a
1438e5 /*
5aØ462 ·* Given a buffer, process a page from it and try to write a corrected page to
26129a ·* the out file. ·Return the number of bytes accessed. ·(Ø if it was unable
6d17aØ ·* to make any corrections.)
a7495d ·*/
4Ø5dØ9 static size_t
2d4b35 DoPage(char const *buf, size_t len, FILE *out, unsigned int page, FILE *log)
31bb36 {
2Ø8fb5 ▷   ParseNode *node;
911fe4 ▷   Heap heap;
a37e1f ▷   HeapCost cost;
738bb8 ▷   OutputHandle oh;
246751 ▷   void (*sighandler)(int);
a4af5a
528b25 ▷   HeapInit(&heap, 1ØØØ);
d5c32c ▷   NodePoolInit();
3ec31d ▷   PerPageInit(buf);
2daf5a
8bc2b3 ▷   /* Initialize signal handling */
127def ▷   interrupt = Ø;
1Ø2de6 ▷   sighandler = signal(SIGINT, inthandler);
b1bccd ▷   if (sighandler != inthandler)
2Ø2c81 ▷   ▷   oldhandler = sighandler;
5eaf5a
4c262a ▷   /* Make a root node */
2bØd91 ▷   node = NodeAlloc();
ff51fb ▷   node->cost = Ø;
d1fa6d ▷   node->refcnt = 1;
99e2b1 ▷   node->input = buf;
9a9bcc ▷   node->subst = &substNull;
2ae768 ▷   ParseStateInit(&node->ps);
756825 ▷   node->parent = NULL;
Ø8af5a
c2fa4d ▷   HeapInsert(&heap, &node->cost);
14af5a
9328Ø7 ▷   /* The main loop: try to extend the current parse. */
7543fc ▷   while ((node = (ParseNode *)HeapGetMin(&heap)) != NULL) {
6a8d77 ▷   ▷   cost = ParseAdvanceString(&heap, node);
f21292 ▷   ▷   if (cost != COST_INFINITY) {
1a952f ▷   ▷   ▷   /* End of the file, or hit a second header line? */
d3443e ▷   ▷   ▷   if (node->input == buf+len ¦¦ InSecondHeader(&node->ps)) {
Øca5f9 ▷   ▷   ▷   ▷   /* Try to wrap up page, if page CRC works */
9b4fa7 ▷   ▷   ▷   ▷   if (node->ps.page_crc == perpage.page_check) {
4Ø3cc2 ▷   ▷   ▷   ▷   ▷   /* Success! */
34a3b2 ▷   ▷   ▷   ▷   ▷   HeapDestroy(&heap);
6bf911 ▷   ▷   ▷   ▷   ▷   OutputInit(&oh, node, NULL);
```

```
a638a2 ▷   ▷   ▷   ▷   ▷   OverstrikeLine("", Ø);
d8af5a
e599c6 ▷   ▷   ▷   ▷   ▷   if (InSecondHeader(&node->ps)) {
c21735 ▷   ▷   ▷   ▷   ▷   ▷   /* Back up to last newline */
afaaØa ▷   ▷   ▷   ▷   ▷   ▷   OutputInit(&oh, node, NULL);
81bØ3c ▷   ▷   ▷   ▷   ▷   ▷   while (OutputGetPrev(&oh) != '\n')
bb4Øde ▷   ▷   ▷   ▷   ▷   ▷   ▷   ;
a5afee ▷   ▷   ▷   ▷   ▷   ▷   OutputUnget(&oh);
971c89 ▷   ▷   ▷   ▷   ▷   }
de79bd ▷   ▷   ▷   ▷   ▷   /* oh points to node that emitted last char on page */
c1c964 ▷   ▷   ▷   ▷   ▷   len = oh.node->input - buf; /* Chars eaten this page */
92675e ▷   ▷   ▷   ▷   ▷   perpage.lines = PrintPage(oh, out);
c9d927 ▷   ▷   ▷   ▷   ▷   DoStats(oh.node, page, log);
6aØ22f ▷   ▷   ▷   ▷   ▷   NodePoolCleanup();
51Ødc6 ▷   ▷   ▷   ▷   ▷   return len;
1e2613 ▷   ▷   ▷   ▷   }
a9Ø42d ▷   ▷   ▷   } else {
84ce96 ▷   ▷   ▷   ▷   /* Keep working on the page */
1bf838 ▷   ▷   ▷   ▷   node->cost = cost += node->cost;
47aa56 ▷   ▷   ▷   ▷   if (node->input > perpage.maxpos) {
67f93d ▷   ▷   ▷   ▷   ▷   perpage.maxpos = perpage.minpos = node->input;
bb9b7a ▷   ▷   ▷   ▷   ▷   if (perpage.max_retries < perpage.retries)
5d2e8b ▷   ▷   ▷   ▷   ▷   ▷   perpage.max_retries = perpage.retries;
97Ø7a1 ▷   ▷   ▷   ▷   ▷   perpage.retries = Ø;▷   /* Made progress */
6c33f6 ▷   ▷   ▷   ▷   } else if (node->input < perpage.minpos) {
f6bfdb ▷   ▷   ▷   ▷   ▷   perpage.minpos = node->input;▷  /* Furthest backtrack */
Øf2613 ▷   ▷   ▷   ▷   }
17Ø359 ▷   ▷   ▷   ▷   ++perpage.retries;
ce84dØ ▷   ▷   ▷   ▷   if (heap.numElems > MAX_HEAP ¦¦ interrupt)
31a3b2 ▷   ▷   ▷   ▷   ▷   HeapDestroy(&heap);
3bfaØ7 ▷   ▷   ▷   ▷   else
Ø35a92 ▷   ▷   ▷   ▷   ▷   AddChildren(node, &heap, buf+len);
b6a3a6 ▷   ▷   ▷   }
a6Øfe4 ▷   ▷   }
7f4824 ▷   ▷   NodeRelease(node);
9e9371 ▷   }
eØ7dØ4 ▷   /* Failed! */
Ødb29c ▷   OverstrikeLine(NULL, Ø);
871Ø51 ▷   puts("Stopping for manual edit.");
91af5a
57396a ▷   NodePoolCleanup();
f8aaØ8 ▷   /* Get rid of the dynamic substitutions */
dØccaa ▷   DoStats(NULL, page, log);
81af5a
fb3e6b ▷   return Ø;
deefe6 }
45af5a
aØ8Øe4 /* The magic file-shuffling routine. */
c3beb7 static int
eØ8df3 RepairFile(char const *name, char const *editor, char const *nastylines)
96bb36 {
499a97 ▷   char buf[PAGE_BUFFER_SIZE];
53dbaØ ▷   char *filename;
627Ød2 ▷   char const *p;
5b77af ▷   size_t namelen;
f1c973 ▷   FILE *in = Ø, *out = Ø, *dump = Ø, *log = Ø;
df6c95 ▷   size_t inbytes;▷▷   /* Bytes in input buffer */
f138fc ▷   size_t outbytes;▷   /* Bytes taken from input buffer */
385Øb8 ▷   unsigned int pages = Ø;▷/* # of pages processed */
cbf5Øe ▷   unsigned int lines = Ø;▷/* # of lines processed (until trouble) */
68f4c6 ▷   unsigned int minline, maxline;▷ /* Where is the error? */
1ece57 ▷   int giveup;▷▷   ▷   /* Have we had to abort corrections? */
fd728e ▷   int err;▷   ▷   ▷   /* Copy of errno for returns */
d2af5a
762783 ▷   globalSize = Ø;▷/* Reset global stats */
4eaf5a
5763e2 ▷   namelen = strlen(name);
b8Ø47Ø ▷   if (!(filename = malloc(namelen+1Ø))) {
e92bØ2 ▷   ▷   p = "Unable to allocate memory\n";
9397ed ▷   ▷   goto error;
439371 ▷   }
eeaf5a
```

```
524332 ▷   memcpy(filename, name, namelen);
2Ø35eØ ▷   strcpy(filename+namelen, ".log");
c2Ø1c2 ▷   puts(filename);
c58b16 ▷   if (!(log = fopen(filename, "at"))) {
27a3d6 ▷   ▷   p = "Unable to open log file \"%s\"\n";
5d97ed ▷   ▷   goto error;
559371 ▷   }
72af5a
2c5fbc ▷   strcpy(filename+namelen, ".out");
a8Ø1c2 ▷   puts(filename);
84c484 ▷   if (!(out = fopen(filename, "at"))) {
efa916 ▷   ▷   p = "Unable to open output file \"%s\"\n";
4997ed ▷   ▷   goto error;
e49371 ▷   }
6eaf5a
Ø1165f retry:
d251ee ▷   /* Read in any new nasty lines */
f8ccea ▷   if (!(in = fopen(nastylines, "rt"))) {
23daec ▷   ▷   fprintf(stderr, "Unable to open nasty lines file \"%s\"\n", nastylines);
12fØ37 ▷   } else {
24414Ø ▷   ▷   ReadNasties(in);
79d358 ▷   ▷   fclose(in);
a59371 ▷   }
6b3545 ▷   /* Try to open input file - .in or original */
7c1b24 ▷   p = filename;
9e8bdØ ▷   strcpy(filename+namelen, ".in");
8992ef ▷   if (!(in = fopen(filename, "rt"))) {
57Ø61b ▷   ▷   if (!(in = fopen(name, "rt"))) {
333d81 ▷   ▷   ▷   filename[namelen] = Ø;
5166c2 ▷   ▷   ▷   p = "Unable to open input file \"%s\"\n";
aØed19 ▷   ▷   ▷   goto error;
67Øfe4 ▷   ▷   }
74Ø2ab ▷   ▷   p = name;
c59371 ▷   }
3d8568 ▷   printf("Repairing from %s\n", p);
d967c6 ▷   strcpy(filename+namelen, ".dmp");
Ød Ødc Ø ▷   if (!(dump = fopen(filename, "wt"))) {
6Øa916 ▷   ▷   p = "Unable to open output file \"%s\"\n";
a997ed ▷   ▷   goto error;
Øe9371 ▷   }
65af5a
5f517c ▷   giveup = Ø;
d953Ø1 ▷   inbytes = Ø;▷   /* Bytes already at the front of the buffer */
9e39Øb ▷   /* Append more data from the file */
fe1b54 ▷   while ((inbytes += fread(buf+inbytes, 1, sizeof(buf)-inbytes, in)) != Ø) {
Ø5a7Ø9 ▷   ▷   if (giveup) {
ca2738 ▷   ▷   ▷   /* Giving up mode - just copy through */
294f45 ▷   ▷   ▷   outbytes = fwrite(buf, 1, inbytes, dump);
3Ø8b1Ø ▷   ▷   ▷   if (!outbytes) {
a6a28a ▷   ▷   ▷   ▷   p = "Error writing dump file!\n";
dc1b7c ▷   ▷   ▷   ▷   goto error;
8ca3a6 ▷   ▷   ▷   }
1f45a6 ▷   ▷   } else {
75c89e ▷   ▷   ▷   outbytes = DoPage(buf, inbytes, out, pages+1, log);
e3cfa3 ▷   ▷   ▷   NodePoolCleanup();
Ø2fØba ▷   ▷   ▷   if (outbytes) {
54999d ▷   ▷   ▷   ▷   pages++;
Ø6d676 ▷   ▷   ▷   ▷   lines += perpage.lines;
b9a12a ▷   ▷   ▷   } else {▷   /* Failed */
532ffØ ▷   ▷   ▷   ▷   /* Find range of backtracking for error location */
Ø43fa5 ▷   ▷   ▷   ▷   minline = 1;
892444 ▷   ▷   ▷   ▷   for (p = buf; ‘p < perpage.minpos; p++)
12a568 ▷   ▷   ▷   ▷   ▷   minline += (*p == '\n');
7136Ø4 ▷   ▷   ▷   ▷   for (maxline = minline; p < perpage.maxpos; p++)
be9ff7 ▷   ▷   ▷   ▷   ▷   maxline += (*p == '\n');
3d1Ø42 ▷   ▷   ▷   ▷   giveup = 1;
Øca3a6 ▷   ▷   ▷   }
Ø6Øfe4 ▷   ▷   }
91bb42 ▷   ▷   /* Fewer bytes now in the buffer */
d6725f ▷   ▷   inbytes -= outbytes;
96bØf2 ▷   ▷   /* Move those bytes to the front again */
611ada ▷   ▷   memmove(buf, buf+outbytes, inbytes);
```

```
a19371 ▷   }
93af5a
f8442b ▷   fclose(in);
1ebdce ▷   in = 0;
2ca9fe ▷   fclose(dump);
6adc3e ▷   dump = 0;
2caf5a
5a1ab5 ▷   /* Okay, let's get tricky */
4c4039 ▷   memcpy(buf, name, namelen);
2a3322 ▷   strcpy(buf+namelen, ".dmp");
c68bd0 ▷   strcpy(filename+namelen, ".in");
1daf5a
41e008 ▷   /* teun: MS Visual C doesn't rename on top of existing file; remove it */
ad887d ▷   if (remove(filename) != 0) {
b1afe5 ▷   ▷   err = errno;
b3ff6a ▷   ▷   fprintf(stderr, "Warning deleting %s\n", filename);
459371 ▷   }
60af5a
3a580c ▷   if (rename(buf, filename) != 0) {
4cafe5 ▷   ▷   err = errno;
6a5bdc ▷   ▷   fclose(out);
934925 ▷   ▷   fclose(log);
b8bf8f ▷   ▷   /* teun: corrected buf, filename order. This cost me an hour */
7398f3 ▷   ▷   fprintf(stderr, "Error renaming %s -> %s\n", buf, filename);
f2c9db ▷   ▷   return err;
f39371 ▷   }
22af5a
c27ad3 ▷   /* This code is spaghetti - is there a cleaner way? */
4d9d93 ▷   if (giveup) {
83acd6 ▷   ▷   printf("Error in %s, lines %u-%u\n", filename, minline, maxline);
dde35d ▷   ▷   fprintf(log, "Error in %s, lines %u-%u\n", filename, minline, maxline);
49a160 ▷   ▷   if (interrupt > 1)
bddfb0 ▷   ▷   ▷   goto manual;
294f75 ▷   ▷   if (editor) {
b2394c ▷   ▷   ▷   if (strcmp(editor, "-") == 0)
f2ebed ▷   ▷   ▷   ▷   goto manual;
683378 ▷   ▷   ▷   sprintf(buf, editor, maxline, filename);
9b45a6 ▷   ▷   } else {
868650 ▷   ▷   ▷   p = getenv("VISUAL");
20c019 ▷   ▷   ▷   if (!p)
01ccee ▷   ▷   ▷   ▷   p = getenv("EDITOR");
65c019 ▷   ▷   ▷   if (!p)
4febed ▷   ▷   ▷   ▷   goto manual;
edf54a ▷   ▷   ▷   sprintf(buf, "%s +%u %s\n", p, maxline, filename);
1a0fe4 ▷   ▷   }
303d32 ▷   ▷   printf("Executing %s\n", buf);
093739 ▷   ▷   globalEdits++;
573724 ▷   ▷   if (system(buf) == 0)
fe9bb9 ▷   ▷   ▷   goto retry;
17e563 ▷   ▷   fputs("Edit failed - aborting\n", stderr);
151e01 manual:
164dad ▷   ▷   puts("Please fix the error by hand and re-run repair.");
239371 ▷   }
ccaf5a
0d1a57 ▷   fclose(out);
e383dd ▷   free(filename);
c6af5a
c9c279 ▷   fprintf(log, "\n%u lines successfully processed.\n", lines);
903b08 ▷   fprintf(log, "Overall substitutions (%u pages):\n", pages);
83ad36 ▷   UseStats(globalStats, log);
5b8f5a ▷   printf("%u manual edits required\n", globalEdits);
e708ae ▷   fclose(log);
71af5a
4a3e6b ▷   return 0;
19af5a
0a60ff error:
55ee6e ▷   err = errno;
e7edff ▷   if (log) fclose(log);
21b8bb ▷   if (dump) fclose(dump);
c26d10 ▷   if (out) fclose(out);
c6def0 ▷   if (in) fclose(in);
a1cb6e ▷   fprintf(stderr, p, filename);
```

```
8283dd ▷   free(filename);
f35ea8 ▷   return err;
4befe6 }
feaf5a
5849cc /* Process the command line, calling RepairFile as needed. */
Øa4d2f int
5Ø8599 main(int argc, char *argv[])
81bb36 {
2512eb ▷   int▷▷   result = Ø;
54e595 ▷   int▷▷   i;
94fce7 ▷   char const *editor = NULL;
9fe87a ▷   char const *nastylines = "nastylines";
Ø1af5a
3b66bd ▷   InitUtil();
18f83c ▷   SubstBuild();
33b89b ▷   memPoolInit(&nastyStructs);
fc2Ø9b ▷   memPoolInit(&nastyStrings);
2aaf5a
882f5c ▷   /* Process leading flags */
917f8f ▷   for (i = 1; i < argc && argv[i][Ø] == '-'; i++) {
2f5bbb ▷   ▷   if (argv[i][1] == '-' && argv[i][2] == Ø) {
dd81ec ▷   ▷   ▷   i++;
eacdcb ▷   ▷   ▷   break;
8Ø91d3 ▷   ▷   } else if (argv[i][1] == 'e') {
653ffØ ▷   ▷   ▷   editor = argv[i][2] ? argv[i]+2 : argv[++i];
63b4aØ ▷   ▷   } else if (argv[i][1] == 'l') {
38Øf25 ▷   ▷   ▷   nastylines = argv[i][2] ? argv[i]+2 : argv[++i];
ca45a6 ▷   ▷   } else {
ac3ffØ ▷   ▷   ▷   editor = argv[i][2] ? argv[i]+2 : argv[++i];
692ce4 ▷   ▷   ▷   fprintf(stderr, "ERROR: Unrecognized option %s\n", argv[i]);
5adb98 ▷   ▷   ▷   return 1;
92Øfe4 ▷   ▷   }
799371 ▷   }
cfaf5a
aac494 ▷   /* Process files */
96e384 ▷   for (; i < argc; i++) {
f74e8a ▷   ▷   result = RepairFile(argv[i], editor, nastylines);
59ce37 ▷   ▷   if (result != Ø) {
96c566 ▷   ▷   ▷   fprintf(stderr, "Fatal error: %s\n", strerror(result));
fddb98 ▷   ▷   ▷   return 1;
43Øfe4 ▷   ▷   }
689371 ▷   }
5baf5a
523e6b ▷   return Ø;
2Øefe6 }
ecaf5a
5938e5 /*
a9e6c5 ·* Local Variables:
939b19 ·* tab-width: 4
3Øe7a4 ·* End:
fØb612 ·* vi: ts=4 sw=4
5c6c42 ·* vim: si
a1495d ·*/
```

```
bc38e5 /*
35aa56 ‘* subst.c -- Repair substitution tables
d3775e ‘*
97a5ec ‘* Copyright (C) 1997 Pretty Good Privacy, Inc.
43775e ‘*
affbf7 ‘* Written by Colin Plumb
84775e ‘*
fad5ed ‘* $Id: subst.c,v 1.14 1997/11/Ø3 22:12:ØØ colin Exp $
77775e ‘*
954b38 ‘* IT IS EXPECTED that users of this program will play with these tables
7a59d7 ‘* and the cost values in the subst.h header. ‘(Some day, they’ll all
721d57 ‘* get moved to an external config file.)
87775e ‘*
d6771Ø ‘* NOTE: Other cost are hiding in the Filter functions in repair.c.
9469c9 ‘* Remember to keep them all on the same scale.
e3495d ‘*/
79af5a
c738e5 /*
d5d88d ‘* The repair program copies its input to its output, making various
fc2feØ ‘* substitutions, until it manages to produce a version that satisfies
3Ø24d3 ‘* the parser. ‘This includes having a correct CRC for each line.
a5cØd4 ‘* Each substitution has a cost, and the combinations are tried in order
1bd3dØ ‘* of increasing cost. ‘NOTE that even translating "A"->"A" counts as
cd6162 ‘* a substitution, although it may have zero cost.
a2775e ‘*
c16667 ‘* The intention is to correct transcription errors, where the
783beb ‘* errors have a distinctly non-uniform distribution. ‘Slight
7db8eØ ‘* differences in cost produce a preference in trying some errors
78ee29 ‘* first. ‘If an error costs half as much as another, combinations
79fd6a ‘* of two of that error will be compared to one of the more expensive.
Ød17e7 ‘* Too many cheap substitutions will result is repair spending
e26163 ‘* a very log time searching before considering the more expensive
aØ92ed ‘* substitutions.
9d775e ‘*
Øe26a5 ‘* The following parameters and the raw substitution tables are expected
a75ab2 ‘* to be edited by the user based on experience. ‘Eventually, this
b68f29 ‘* will be moved into an external config file, but for now it’s a matter
79ffa2 ‘* of recompiling.
55495d ‘*/
edaf5a
c9dØ4a #include "subst.h"
7849Øa #include "util.h"
5eaf5a
8bØØeØ /* what the OCR software reports for "unrecognizable */
969389 #define UNRECOG␣STRING "~\274"
8eaf5a
b538e5 /*
97843Ø ‘* The input substitutions to make (one-to-one). ‘‘These are listed in
e65cb9 ‘* the order of correction. i.e. uncorrected input first, then corrected
29e44b ‘* output. ‘Substitutions are one-way; to get two-way, list it twice.
Øe495d ‘*/
1aaf5a
e31ab5 struct RawSubst const substSingles[] = {
947446 ▷       /* Identity substitutions - note that period (.) is excluded */
5Øf0fd ▷       { "!\"#$%&’()*+,-./Ø123456789:;<=>?" SPACE␣STRING,
eb2536 ▷       ‘‘"!\"#$%&’()*+,-./Ø123456789:;<=>?" SPACE␣STRING, Ø, Ø, NULL },
4ea632 ▷       { "@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\\]^␣\t" TAB␣STRING,
62f1c5 ▷       ‘‘"@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\\]^␣\t" TAB␣STRING, Ø, Ø, NULL },
c46712 ▷       { "‘abcdefghijklmnopqrstuvwxyz{¦}~\f" FORMFEED␣STRING,
e77caa ▷       ‘‘"‘abcdefghijklmnopqrstuvwxyz{¦}~\f" FORMFEED␣STRING, Ø, Ø, NULL },
2Ø4bf9 #if (TAB␣PAD␣CHAR & 128)▷       /* Not already included? */
a95a32 ▷       { TAB␣PAD␣STRING, TAB␣PAD␣STRING, Ø, NULL },
6a7454 #endif
37eØØ1 ▷       { "\r\n" CONTIN␣STRING, "\n\n" CONTIN␣STRING, Ø, Ø, NULL },
3daf5a
91b473 ▷       /* Occasionally these just get inserted as glitches */
8Ø58ca ▷       { ".,’‘", NULL, 5, 1Ø, FilterNearBlanks },
fdaØ27 ▷       /* This is now pretty infrequent */
Ø3ba67 ▷       { "-␣", "␣-", Ø, 1Ø, FilterAfterRepeat },
eeaf5a
96f27c ▷       /*
ba468a ▷       ‘* Capitalization errors are common in some cases
```

```
197912 ▷       ‘* c/C, s/S, u/U are fucked up all the time.
b2ebØd ▷       ‘* Also o/O, v/V and w/W. ‘x, y and z also give some problems.
aØ4Ødf ▷       ‘*/
1bf95d ▷       { "cilmopsuvwxyz", "CILMOPSUVWXYZ", 7, 13, FilterNearLower },
f3867Ø ▷       { "CILMOPSUVWXYZ", "cilmopsuvwxyz", 7, 13, FilterNearUpper },
b71cØb ▷       /* Other errors */
7bbf2a ▷       { "g9aaiji;xXØØSi", "9gg2ji;i%%oO3f", 1Ø, Ø, NULL },
332e4f ▷       /* This seems to happen a lot */
Ø8aa9b ▷       { "c", "r", 9, Ø, NULL },
3daf5a
3588f9 ▷       { "j", ";", 9, Ø, NULL },
b28659 ▷       { "’ ", "‘‘", 1Ø, Ø, NULL },
d5af5a
11d1d2 ▷       /* Uncommon errors */
f5af5a
Øea195 ▷       /* Wierd stuff that’s happened in the checksum part */
8c5Ø6b ▷       /* A highish weight is okay here */
2c73f3 ▷       { "sSEdJl", "554437", 15, Ø, NULL },
635bc3 ▷       { "LESsPZ", "bb8a22", 15, Ø, NULL },
35af5a
2d5459 ▷       /* Wierd stuff that has happened */
e153ee ▷       { "BasAeaeRoooo", "3334a@QQpqbd", 5, 15, FilterIsBinary },
cbfe8b ▷       { "oooo", "pqbd", Ø, 15, FilterIsBinary },
9Ødc94 ▷       { "ttTCCflO", "iff{[lfG", 12, Ø, NULL },
6fb177 #if Ø
59f3a9 ▷       /* If the line-breaks get screwed up, use these */
6fØb59 ▷       { " ", "\n", 1Ø, COST_INFINITY, FilterChecksumFollows },
dØ3133 ▷       { "\n", " ", COST_INFINITY, 1Ø, FilterChecksumFollows },
9Øc867 ▷       { "\n", NULL, COST_INFINITY , 11, FilterChecksumFollows },
967454 #endif
2Øaf5a
45b6e9 { NULL, NULL, Ø, Ø, NULL }
9b82f7 };
a2af5a
b359c8 /* The many-to-many substitutions */
5fe242 struct RawSubst const substMultiples[] = {
24fd52 ▷       { "’’", "\"", 2, Ø, NULL },
4e1e59 ▷       { "‘‘", "\"", 2, Ø, NULL },
33d2Ø7 ▷       { ",’", "\"", 2, Ø, NULL },
8ada9c ▷       { "’,", "\"", 2, Ø, NULL },
7af5c9 ▷       { ",,", "\"", 2, Ø, NULL },
bc3aØ2 ▷       /* Extra inserted spaces are common */
25d51a ▷       { " ", " ", COST_INFINITY, ‘Ø, FilterFollowsSpace },
e343da ▷       { " ", "", Ø, 15, FilterFollowsSpace },
873f61 ▷       { "\t", " ", COST_INFINITY, ‘Ø, FilterFollowsSpace },
9fa244 ▷       { "\t", "", Ø, 1Ø, FilterFollowsSpace },
949372 ▷       /* Convert between SPACE_CHAR dots and periods */
69b66b ▷       { ".", SPACE_STRING, 1, COST_INFINITY, FilterFollowsSpace },
d1518d ▷       { ".", " "SPACE_STRING, COST_INFINITY, 1Ø, FilterFollowsSpace },
352338 ▷       { SPACE_STRING, ".", 15, 5, FilterFollowsSpace },
Ø9acab ▷       { SPACE_STRING, " "SPACE_STRING, COST_INFINITY, 5, FilterFollowsSpace },
2faf5a
4c3cØ5 ▷       /* Replace "unknown" by zero - it often is */
fab1d6 ▷       { UNRECOG_STRING, "Ø", 1, Ø, NULL },
ee1Ø3Ø ▷       { UNRECOG_STRING, "␣", 2, Ø, NULL },
f2bd25 ▷       { UNRECOG_STRING, ")", 3, Ø, NULL },
Ø6e6ba ▷       { UNRECOG_STRING, "^", 4, Ø, NULL },
fba3aØ ▷       /* Except that these glitches are common */
61ca71 ▷       { UNRECOG_STRING"’", "\\\"", Ø, Ø, NULL },
1162de ▷       { UNRECOG_STRING"’", "\"", 1, Ø, NULL },
d42139 ▷       { "’"UNRECOG_STRING, "\"", Ø, Ø, NULL },
7e983c ▷       { UNRECOG_STRING UNRECOG_STRING , "\"", Ø, Ø, NULL },
9d356b ▷       /* Something else that has been seen */
d5eØ72 ▷       { "V’", "\\\"", 5, Ø, NULL },
63af5a
db2aaa ▷       /* A common transposition */
7c42f2 ▷       { "\"’", "’\"", 5, Ø, NULL },
b22c55 ▷       { "’\"", "\"’", 5, Ø, NULL },
Øa85f9 ▷       /* These also happen fairly often */
23d7ff ▷       { " \"", "’’", 5, Ø, NULL },
23e376 ▷       { "\" ", "’’", 5, Ø, NULL },
8daf5a
```

```
0b0995 ▷        /* Common glitches */
0b29d2 ▷        { "\t.\n", "\n", 5, 0, NULL },
5bbb63 ▷        { "\t,\n", "\n", 5, 0, NULL },
037633 ▷        { "\t-\n", "\n", 5, 0, NULL },
11f5d5 ▷        { "\t␣\n", "\n", 5, 0, NULL },
9cbe64 ▷        { "\t'\n", "\n", 5, 0, NULL },
7d0be4 ▷        { "\t`\n", "\n", 5, 0, NULL },
a55b0c ▷        { "\t~\n", "\n", 5, 0, NULL },
a5b16d ▷        { "\t:\n", "\n", 5, 0, NULL },
d5a6bd ▷        { "\t"SPACE␣STRING"\n", "\n", 5, 0, NULL },
8faf5a
d929be ▷        /* Less common */
10fa92 ▷        { " .\n", "\n", 10, 0, NULL },
415e02 ▷        { " ,\n", "\n", 10, 0, NULL },
fd0c4a ▷        { " -\n", "\n", 10, 0, NULL },
5a8ef1 ▷        { " ␣\n", "\n", 10, 0, NULL },
c52ab8 ▷        { " '\n", "\n", 10, 0, NULL },
b0ae05 ▷        { " `\n", "\n", 10, 0, NULL },
21c513 ▷        { " ~\n", "\n", 10, 0, NULL },
40b776 ▷        { " :\n", "\n", 10, 0, NULL },
64b5b6 ▷        { " "SPACE␣STRING"\n", "\n", 10, 0, NULL },
adaf5a
938eeb ▷        /* Even less common */
4fb6df ▷        { ".\n", "\n", 15, 0, NULL },
fd124f ▷        { ",\n", "\n", 15, 0, NULL },
364007 ▷        { "-\n", "\n", 15, 0, NULL },
bbc2bc ▷        { "␣\n", "\n", 15, 0, NULL },
6466f5 ▷        { "'\n", "\n", 15, 0, NULL },
aee248 ▷        { "`\n", "\n", 15, 0, NULL },
be895e ▷        { "~\n", "\n", 15, 0, NULL },
f8fb3b ▷        { ":\n", "\n", 15, 0, NULL },
7a52aa ▷        { SPACE␣STRING"\n", "\n", 15, 0, NULL },
66af5a
cb5459 ▷        /* Wierd stuff that has happened */
f7c67f ▷        { "lJ", "U", 10, 0, NULL },
2125cd ▷        { "ll", "U", 10, 0, NULL },
4e2de9 ▷        { "l1", "U", 10, 0, NULL },
91bab9 ▷        { "il", "U", 10, 0, NULL },▷    /* Fairly common, actually */
8e1b88 ▷        { "li", "U", 10, 0, NULL },
4b7bfe ▷        { "l)", "U", 10, 0, NULL },
20aaad ▷        { "Ll", "U", 10, 0, NULL },
93a4d3 ▷        { "LI", "U", 10, 0, NULL },
d4a289 ▷        { "L1", "U", 10, 0, NULL },
13af5a
b0acd6 ▷        { "lo", "b", 10, 0, NULL },
dd0c26 ▷        { "cl", "d", 10, 0, NULL },
ba6f97 ▷        { "cliff", "diff", 2, 0, NULL },
0dbbb0 ▷        { "*\n", "*/\n", 10, 0, NULL },
64af5a
a04b89 ▷        /* That big black block has odd things happen to it */
c3807c ▷        { "d", CONTIN␣STRING, 10, 0, NULL },
fd6989 ▷        { "d\n", CONTIN␣STRING"\n", 3, 0, NULL },
3863f1 ▷        { "S", CONTIN␣STRING, 10, 0, NULL },
7407eef ▷       { "S\n", CONTIN␣STRING"\n", 3, 0, NULL },
05af5a
c67bb6 ▷        /* Tab-stop wonders */
4d4565 ▷        { TAB␣STRING, TAB␣STRING"", 0, 0, TabFilter },
4d2f6d ▷        { TAB␣STRING, TAB␣STRING" ", 0, 0, TabFilter },
c20abd ▷        { TAB␣STRING, TAB␣STRING" ·", 0, 0, TabFilter },
fadc15 ▷        { TAB␣STRING, TAB␣STRING" ··", 0, 0, TabFilter },
95f581 ▷        { TAB␣STRING, TAB␣STRING" ···", 0, 0, TabFilter },
422705 ▷        { TAB␣STRING, TAB␣STRING" ····", 0, 0, TabFilter },
72e5fb ▷        { TAB␣STRING, TAB␣STRING" ·····", 0, 0, TabFilter },
c2fbc8 ▷        { TAB␣STRING, TAB␣STRING" ·····", 0, 0, TabFilter },
2961f5 ▷        /* Some scan errors */
2256cb ▷        { "D ", TAB␣STRING"", 1, 5, TabFilter },
2dd954 ▷        { "D ", TAB␣STRING" ", 1, 5, TabFilter },
f51c57 ▷        { "D ", TAB␣STRING" ·", 1, 5, TabFilter },
032e09 ▷        { "D ", TAB␣STRING" ··", 1, 5, TabFilter },
8f95c0 ▷        { "D ", TAB␣STRING" ···", 1, 5, TabFilter },
abceb6 ▷        { "D ", TAB␣STRING" ····", 1, 5, TabFilter },
d8d85c ▷        { "D ", TAB␣STRING" ·····", 1, 5, TabFilter },
```

```
--7ae1 ØØ133a3238d8ØØ2ØØØd Page 4 of subst.c

a39Ø1e ▷       { "D ", TAB␣STRING" ^^^^^^", 1, 5, TabFilter },
af8ef7 #if TAB␣PAD␣CHAR != ' '
459261 #error Fix those tab patterns!
1e7454 #endif
e4b6e9 { NULL, NULL, Ø, Ø, NULL }
3382f7 };
```

```
bc38e5 /*
ce1eec ·* subst.h -- Header for repair substitutions
d9775e ·*
84a5ec ·* Copyright (C) 1997 Pretty Good Privacy, Inc.
a2775e ·*
73fbf7 ·* Written by Colin Plumb
8a775e ·*
7374fØ ·* $Id: subst.h,v 1.9 1997/11/Ø3 22:12:ØØ colin Exp $
42495d ·*/
faaf5a
5338e5 /*
Øa493a ·* Give up if the list of pending changes to attempt grows to this many
ØØ3525 ·* elements. ·Each element is 32 bytes, so 128K is 8 MB of memory.
42cb44 ·* (Other than this, repair's memory usage is fairly modest.)
15495d ·*/
4d47fd #define MAX_HEAP (1<<17)
Ø7af5a
7338e5 /*
4457da ·* There is a hack in the code to find a single substitution that will fix a
a239fb ·* line, even if it's not in the tables. ·It gets added to the tables "on
7be8c2 ·* probation", with an infinite cost, and if it leads to a successful
751e3c ·* correction of the entire page, is "learned" for future use and its
325d9b ·* cost reduced to something finite.
622afd ·* (This is not remembered across runs of the program, though.
7c3fce ·* Edit the tables in the source to fix it.)
eb495d ·*/
12824Ø #define DYNAMIC_COST_LEARNED 15
abaf5a
b438e5 /*
3975c5 ·* This negative-cost bonus for passing the end of a line with the right
66aØ9d ·* CRC makes the search engine reluctant to backtrack past a correct CRC,
cc4983 ·* greatly improving efficiency. ·It's rather a hack, though. ·Think of
4982Ø9 ·* this in terms of "how many errors should be considered in the current
b56fc2 ·* line before considering the possibility of errors in the previous line?"
9d775e ·*
2a3886 ·* This bonus is halved for lines that are the result of a correction
72c7e7 ·* that was computed from the checksum, since a correct checksum is
a8a9df ·* much less significant in such a case.
51495d ·*/
1c298b #define COST_LINE -3Ø
e2af5a
61ec1b /* The cost of a full-line nastyline substitution. */
4Øf Ø3e #define NASTY_COST 5
86af5a
d3fØØe /* Type describing filter functions used in substitutions */
f196c7 struct ParseNode;
a71df2 struct Substitution;
b26Ø9b #include "heap.h"
a124f7 typedef HeapCost FilterFunc(struct ParseNode *parent, char const *limit,
f35244 ▷       struct Substitution const *subst);
96642d FilterFunc TabFilter, ···········FilterFollowsSpace, FilterNearBlanks;
8636ea FilterFunc FilterNearUpper, ·····FilterNearLower, ···FilterNearXDigit;
5cd3ef FilterFunc FilterAfterRepeat, ····FilterCharConst, ···FilterChecksumFollows;
371f68 FilterFunc FilterLikelyUnderscore, FilterIsDynamic, ···FilterIsBinary;
76af5a
4f44e3 /* The external substitution format */
ad2eb9 typedef struct RawSubst {
bØ9721 ▷       char const *input;
868834 ▷       char const *output;
cØacda ▷       HeapCost cost, cost2;
2c24ad ▷       FilterFunc *filter;
e24f87 } RawSubst;
aØaf5a
f1a94f /* The substitutions to make */
84a82Ø extern struct RawSubst const substSingles[];
65f149 extern struct RawSubst const substMultiples[];
```

```
bc38e5 /*
366Ø1Ø ·* unmunge.c -- Program to convert a munged file to original form
c4775e ·*
4ca5ec ·* Copyright (C) 1997 Pretty Good Privacy, Inc.
5e775e ·*
dbbd99 ·* Designed by Colin Plumb, Mark H. Weaver, and Philip R. Zimmermann
997659 ·* Written by Mark H. Weaver
fd775e ·*
11f1Øb ·* $Id: unmunge.c,v 1.13 1997/11/13 23:27:Ø8 mhw Exp $
4b495d ·*/
bbaf5a
41d4f4 #include <sys/stat.h>
5ef15e #include <sys/types.h>
Øa37bc #include <fcntl.h>
a217c2 #include <unistd.h>
3baf5a
fØede3 /*#include <direct.h> ··teun: MS VC wants direct.h for mkdir */
6baf5a
cafeb2 #include <stdio.h>
78495d #include <errno.h>
ae324c #include <string.h>
14b1cb #include <ctype.h>
5fbea3 #include <stdlib.h>
Øfbb5f #include <assert.h>
d7af5a
f949Øa #include "util.h"
Øcaf5a
bda399 typedef struct UnMungeState
62bb36 {
dbf5e4 ▷   char const *▷   mungedFileName;
acØ714 ▷   char▷    ▷    ▷   dirName[128];
4ea62e ▷   char▷    ▷    ▷   fileName[128];
86b5f9 ▷   char *▷  ▷    ▷   fileNameTail;
a67ffa ▷   int▷▷    ▷    ▷   binaryMode, tabWidth;
3c7744 ▷   long▷    ▷    ▷   productNumber, fileNumber, pageNumber, lineNumber;
beb1a6 ▷   long▷    ▷    ▷   manifestLineNumber;
b99ad5 ▷   word16▷  ▷    ▷   hdrFlags;
Ø2d466 ▷   CRC▷▷    ▷    ▷   pageCRC, seenPageCRC;
2Ø3a7a ▷   FILE *▷  ▷    ▷   manifest;
bd237d ▷   FILE *▷  ▷    ▷   file;
Ø54Ø43 ▷   FILE *▷  ▷    ▷   out;
d43dc6 } UnMungeState;
14af5a
79af5a
dd53a9 /* Returns number of characters decoded, or -1 on error */
12beb7 static int
fc3fb1 Decode4(char const src[4], byte dest[3])
f4bb36 {
9c5521 ▷   int▷▷   i, length;
Ød9Øbd ▷   byte▷   srcVal[4];
eeaf5a
79d7ef ▷   for (i = Ø; i < 4 && src[i] != RADIX64_END_CHAR; i++)
aea84a ▷   ▷   if ((srcVal[i] = Radix64DigitValue(src[i])) == (byte) -1)
2Ødb98 ▷   ▷   ▷   return 1;
6Øaf5a
ec7f6b ▷   length = i - 1;
Ødedec ▷   if (length < 1)
acØ43c ▷   ▷   return -1;
53af5a
97dc1e ▷   for (; i < 4; i++)
d2a324 ▷   ▷   srcVal[Ø] = Ø;
f8af5a
bb8d5Ø ▷   dest[Ø] = (srcVal[Ø] << 2) ¦ (srcVal[1] >> 4);
ecc5bb ▷   dest[1] = (srcVal[1] << 4) ¦ (srcVal[2] >> 2);
fc5e14 ▷   dest[2] = (srcVal[2] << 6) ¦ (srcVal[3]);
a9af5a
Ø6e5ee ▷   return length;
41efe6 }
58af5a
1e38e5 /*
2c53ad ·* Return number of characters decoded, or -1 on error
cb495d ·*/
```

```
Ø5beb7 static int
eba565 DecodeLine(char const *src, char *dest, int srclength)
a2bb36 {
Ø94c9a ▷   int destlength = Ø;
15da6Ø ▷   int result;
cØaf5a
Øa9ef8 ▷   if (srclength % 4 ¦¦ !srclength)
Ø2c8e5 ▷   ▷   return -1;▷ /* Must be a multiple of 4 */
71af5a
918b47 ▷   while (srclength -= 4) {
6396e8 ▷   ▷   if (Decode4(src, dest + destlength) != 3)
963fd5 ▷   ▷   ▷   return -1;
Ø2aa48 ▷   ▷   src += 4;
6d8f95 ▷   ▷   destlength += 3;
c69371 ▷   }
78b1e9 ▷   result = Decode4(src, dest + destlength);
12f7Ø6 ▷   if (result < 1)
84Ø43c ▷   ▷   return -1;
24e85d ▷   return destlength + result;
58efe6 }
1daf5a
1Øef99 int PrintFileError(UnMungeState *state, char const *message)
2dbb36 {
727ae9 ▷   fprintf(stderr, "%s, %s line %ld\n", message,
a12Øc1 ▷   ▷   ▷   state->mungedFileName, state->lineNumber);
1564b7 ▷   return 1;
8eefe6 }
f6af5a
f12367 int ReadManifest(UnMungeState *state, long fileNumberWanted,
6c17aØ ▷   ▷   ▷   ▷   ·char const *fileTailPrefix, long prefixLen)
38bb36 {
5Øc3Ø8 ▷   long▷   ▷   fileNumber = Ø;
da569a ▷   long▷   ▷   firstMissingFileNum = Ø, lastMissingFileNum = Ø;
126997 ▷   char▷   ▷   buffer[512];
31Ø58b ▷   char *▷ ▷   p;
2aaf5a
43c692 ▷   if (state->manifest == NULL)
68c7a1 ▷   {
647fd6 ▷   ▷   if (fileNumberWanted != Ø)
a25b34 ▷   ▷   {
adØ7Ø7 ▷   ▷   ▷   assert(fileTailPrefix != NULL);
559765 ▷   ▷   ▷   strncpy(state->fileName, fileTailPrefix, sizeof(state->fileName));
dbdd6a ▷   ▷   ▷   state->fileName[sizeof(state->fileName) - 1] = '\Ø';
b7c963 ▷   ▷   ▷   state->fileNameTail = state->fileName;
ae0fe4 ▷   ▷   }
c7bbde ▷   ▷   return Ø;
Ø59371 ▷   }
58e748 ▷   while (fgets(buffer, sizeof(buffer), state->manifest))
Øcc7a1 ▷   {
96daab ▷   ▷   if ((p = strchr(buffer, '\n')) != NULL)
d8Ø4fe ▷   ▷   ▷   *p = '\Ø';
57e1c7 ▷   ▷   state->manifestLineNumber++;
3e66e2 ▷   ▷   if (buffer[Ø] == 'D')
885b34 ▷   ▷   {
aØ8f69 ▷   ▷   ▷   if (buffer[1] != ' ')
18bb99 ▷   ▷   ▷   ▷   goto invalidManifest;
c79f93 ▷   ▷   ▷   strncpy(state->dirName, buffer + 2, sizeof(state->dirName));
bbaa7a ▷   ▷   ▷   if (state->dirName[sizeof(state->dirName) - 1] != '\Ø')
9bbb99 ▷   ▷   ▷   ▷   goto invalidManifest;
76Øfe4 ▷   ▷   }
16Øe1d ▷   ▷   else
495b34 ▷   ▷   {
2d6f63 ▷   ▷   ▷   fileNumber = strtol(buffer, &p, 1Ø);
4231d1 ▷   ▷   ▷   if (p == buffer ¦¦ *p != ' ')
b9bb99 ▷   ▷   ▷   ▷   goto invalidManifest;
b664c8 ▷   ▷   ▷   p++;
f8af5a
6bab36 ▷   ▷   ▷   if (fileNumberWanted == Ø ¦¦ fileNumber < fileNumberWanted)
c7f776 ▷   ▷   ▷   {
d1dda5 ▷   ▷   ▷   ▷   if (firstMissingFileNum == Ø)
22caab ▷   ▷   ▷   ▷   ▷   firstMissingFileNum = fileNumber;
6662a2 ▷   ▷   ▷   ▷   lastMissingFileNum = fileNumber;
```

```
1bdØ4Ø ▷   ▷   ▷   ▷   continue;
faa3a6 ▷   ▷   ▷   }
1Ø2f74 ▷   ▷   ▷   else if (fileNumber > fileNumberWanted)
daf622 ▷   ▷   ▷   ▷   break;
96bb8c ▷   ▷   ▷   else
77f776 ▷   ▷   ▷   {
addf91 ▷   ▷   ▷   ▷   size_t▷ ▷   len;
cdaf5a
dØ8aad ▷   ▷   ▷   ▷   len = strlen(state->dirName);
18bbcØ ▷   ▷   ▷   ▷   assert(sizeof(state->fileName) >= sizeof(state->dirName));
d86454 ▷   ▷   ▷   ▷   memcpy(state->fileName, state->dirName, len);
459a56 ▷   ▷   ▷   ▷   strncpy(state->fileName + len, p,
fac389 ▷   ▷   ▷   ▷   ▷   ▷   sizeof(state->fileName) - len);
9bf2a4 ▷   ▷   ▷   ▷   if (strncmp(p, fileTailPrefix, prefixLen) != Ø)
Øc72c3 ▷   ▷   ▷   ▷   {
cc445c ▷   ▷   ▷   ▷   ▷   fprintf(stderr, "Mismatched filename, headers say '%s',\n"
9258b2 ▷   ▷   ▷   ▷   ▷   ▷   ▷   " ‘manifest says '%s'\n",
a135e6 ▷   ▷   ▷   ▷   ▷   ▷   ▷   fileTailPrefix, p);
9a9579 ▷   ▷   ▷   ▷   ▷   return 1;
bb2613 ▷   ▷   ▷   ▷   }
bØbf59 ▷   ▷   ▷   ▷   p = state->dirName;
e12Øb0 ▷   ▷   ▷   ▷   while ((p = strchr(p, '/')) != NULL)
5d72c3 ▷   ▷   ▷   ▷   {
65f237 ▷   ▷   ▷   ▷   ▷   *p = '\Ø';
5ebd8b ▷   ▷   ▷   ▷   ▷   mkdir(state->dirName, Ø777);
5Øea3c ▷   ▷   ▷   ▷   ▷   *p++ = '/';
a82613 ▷   ▷   ▷   ▷   }
32fb7e ▷   ▷   ▷   ▷   state->fileNameTail = state->fileName + len;
3Øf622 ▷   ▷   ▷   ▷   break;
f5a3a6 ▷   ▷   ▷   }
48Øfe4 ▷   ▷   }
629371 ▷   }
db44b9 ▷   if (firstMissingFileNum != Ø)
29c7a1 ▷   {
2e14e9 ▷   ▷   fprintf(stderr, "Missing files %ld-%ld\n",
Ø5a822 ▷   ▷   ▷   ▷   firstMissingFileNum, lastMissingFileNum);
6a9371 ▷   }
a9e8Ø2 ▷   if (fileNumberWanted != Ø && fileNumber != fileNumberWanted)
dcc7a1 ▷   {
abdfba ▷   ▷   fprintf(stderr, "Can't find file %ld in manifest file\n",
Øaca15 ▷   ▷   ▷   ▷   fileNumberWanted);
b3e1Ø2 ▷   ▷   return 1;
929371 ▷   }
b43e6b ▷   return Ø;
8baf5a
22b688 invalidManifest:
d73e59 ▷   fprintf(stderr, "Error parsing manifest file, line %ld\n",
a53cfa ▷   ▷   ▷   state->manifestLineNumber);
4b64b7 ▷   return 1;
47efe6 }
cdaf5a
51a9c7 int UnMungeFile(char const *mungedFileName, char const *manifestFileName,
aØb4d5 ▷   ▷   ▷   ▷   int forceOverwrite, int forcePartialFiles)
b8bb36 {
dae1Ø9 ▷   UnMungeState *▷ state;
45d86f ▷   EncodeFormat const *▷   fmt = NULL;
cØd775 ▷   char▷   ▷   ▷   buffer[512];
ac41aa ▷   char▷   ▷   ▷   outbuf[BYTES_PER_LINE+1];
196Ø36 ▷   char *▷ ▷   ▷   line;
b6969a ▷   char *▷ ▷   ▷   lineData;
6cfe15 ▷   char *▷ ▷   ▷   p;
77Øcf6 ▷   int▷▷   ▷   ▷   length;
4ecfe7 ▷   int▷▷   ▷   ▷   result = Ø;
3fcaØb ▷   int▷▷   ▷   ▷   skipPage = Ø;
45fdfc ▷   CRC▷▷   ▷   ▷   lineCRC;
4eaf8e ▷   word32▷ ▷   ▷   num;
e9af5a
c45d21 ▷   state = (UnMungeState *)calloc(1, sizeof(*state));
aØ4afc ▷   state->mungedFileName = mungedFileName;
d9af5a
fbd5e6 ▷   if (manifestFileName != NULL)
65c7a1 ▷   {
```

```
2fc4d7 ▷   ▷   if ((state->manifest = fopen(manifestFileName, "r")) == NULL)
Ø6e51b ▷   ▷   ▷   goto errnoError;
b39371 ▷   }
eaaf5a
4Ø1671 ▷   if ((state->file = fopen(state->mungedFileName, "r")) == NULL)
7fd146 ▷   ▷   goto errnoError;
c2af5a
da1978 ▷   while (!feof(state->file))
bac7a1 ▷   {
f451a5 ▷   ▷   if (fgets(buffer, sizeof(buffer), state->file) == NULL)
6e5b34 ▷   ▷   {
8244d8 ▷   ▷   ▷   if (feof(state->file))
15f622 ▷   ▷   ▷   ▷   break;
c96521 ▷   ▷   ▷   goto fileError;
3ØØfe4 ▷   ▷   }
d5af5a
d1fe43 ▷   ▷   state->lineNumber++;
a9af5a
798693 ▷   ▷   line = buffer;
9d3b55 ▷   ▷   /* Strip leading whitespace */
Ø99584 ▷   ▷   while (isspace(*line))
441863 ▷   ▷   ▷   line++;
1a9c24 ▷   ▷   if (*line == '\Ø')
5517f5 ▷   ▷   ▷   continue;
1caf5a
a213a1 ▷   ▷   /* Strip trailing whitespace */
4a7239 ▷   ▷   p = line + strlen(line);
b7dfØa ▷   ▷   while (p > line && (byte)p[-1] < 128 && isspace(p[-1]))
44f98b ▷   ▷   ▷   p--;
cØaf5a
1c95c3 ▷   ▷   lineData = line + PREFIX_LENGTH;
13af5a
21ff33 ▷   ▷   /* Pad up to at least PREFIX_LENGTH */
cØd356 ▷   ▷   while (p < lineData)
18Ø3fe ▷   ▷   ▷   *p++ = ' ';
73Ø222 ▷   ▷   *p++ = '\n';
513f17 ▷   ▷   *p = '\Ø';
fd2e54 ▷   ▷   length = p - lineData;
8faf5a
25b813 ▷   ▷   if (line[Ø] == HDR_PREFIX_CHAR)
e25b34 ▷   ▷   {
b35fff ▷   ▷   ▷   fmt = FindFormat(line[1]);
e1a4b4 ▷   ▷   ▷   if (!fmt)
6df776 ▷   ▷   ▷   {
6e12Øf ▷   ▷   ▷   ▷   result = PrintFileError(state, "ERROR: Invalid header type");
621b7c ▷   ▷   ▷   ▷   goto error;
8fa3a6 ▷   ▷   ▷   }
f1Øfe4 ▷   ▷   }
6daf5a
42e654 ▷   ▷   lineCRC = CalculateCRC(fmt->lineCRC, Ø, (byte const *)lineData, length);
5caf5a
6ad77Ø ▷   ▷   p = line + EncodedLength(fmt, fmt->runningCRCBits);
5a7f6Ø ▷   ▷   if (DecodeCheckDigits(fmt, p, NULL, fmt->lineCRC->bits, &num)
36bcc1 ▷   ▷   ▷   ¦¦ lineCRC != num)
Øb5b34 ▷   ▷   {
3dea99 ▷   ▷   ▷   result = PrintFileError(state, "ERROR: Line CRC failed");
dded19 ▷   ▷   ▷   goto error;
ccØfe4 ▷   ▷   }
2aaf5a
24b813 ▷   ▷   if (line[Ø] == HDR_PREFIX_CHAR)
f85b34 ▷   ▷   {
f747b9 ▷   ▷   ▷   int▷▷   ▷   formatVersion;
6a1d41 ▷   ▷   ▷   int▷▷   ▷   flags;
ec79a7 ▷   ▷   ▷   CRC▷▷   ▷   seenPageCRC;
411cef ▷   ▷   ▷   int▷▷   ▷   tabWidth;
e7be38 ▷   ▷   ▷   long▷   ▷   productNumber;
c92d6c ▷   ▷   ▷   long▷   ▷   fileNumber;
db9e93 ▷   ▷   ▷   long▷   ▷   pageNumber;
99572b ▷   ▷   ▷   char *▷ ▷   fileNameTail;
7Ø56e5 ▷   ▷   ▷   int▷▷   ▷   skipNextPage = Ø;
bcdc83 ▷   ▷   ▷   char *▷ ▷   p;
816747 ▷   ▷   ▷   EncodeFormat const *▷   hFmt = &hexFormat;
```

```
e0af5a
853f8b ▷   ▷   ▷   /* Parse header line */
3e29ce ▷   ▷   ▷   p = lineData;
62af5a
53c7ba ▷   ▷   ▷   if (DecodeCheckDigits(hFmt, p, &p, HDR_VERSION_BITS, &num))
72f776 ▷   ▷   ▷   {
85f50a ▷   ▷   ▷   invalidHeader:
c1e53a ▷   ▷   ▷   ▷   result = PrintFileError(state, "ERROR: Invalid header");
601b7c ▷   ▷   ▷   ▷   goto error;
9ca3a6 ▷   ▷   ▷   }
4c680a ▷   ▷   ▷   formatVersion = num;
7faf5a
eb1916 ▷   ▷   ▷   if (DecodeCheckDigits(hFmt, p, &p, HDR_FLAG_BITS, &num))
1cdda2 ▷   ▷   ▷   ▷   goto invalidHeader;
1abe9c ▷   ▷   ▷   flags = num;
23af5a
4967b6 ▷   ▷   ▷   if (DecodeCheckDigits(hFmt, p, &p, fmt->pageCRC->bits, &num))
e2dda2 ▷   ▷   ▷   ▷   goto invalidHeader;
3d881b ▷   ▷   ▷   seenPageCRC = num;
7aaf5a
2f7793 ▷   ▷   ▷   if (DecodeCheckDigits(hFmt, p, &p, HDR_TABWIDTH_BITS, &num))
f5dda2 ▷   ▷   ▷   ▷   goto invalidHeader;
6ca041 ▷   ▷   ▷   tabWidth = num;
35af5a
caac93 ▷   ▷   ▷   if (DecodeCheckDigits(hFmt, p, &p, HDR_PRODNUM_BITS, &num))
a3dda2 ▷   ▷   ▷   ▷   goto invalidHeader;
ffd423 ▷   ▷   ▷   productNumber = num;
43af5a
9c0aa5 ▷   ▷   ▷   if (DecodeCheckDigits(hFmt, p, &p, HDR_FILENUM_BITS, &num))
a0dda2 ▷   ▷   ▷   ▷   goto invalidHeader;
948323 ▷   ▷   ▷   fileNumber = num;
24af5a
955f0f ▷   ▷   ▷   if (sscanf(p, " Page %ld of ", &pageNumber) < 1)
67dda2 ▷   ▷   ▷   ▷   goto invalidHeader;
edaf5a
db4b6e ▷   ▷   ▷   if (formatVersion > 0)
32f776 ▷   ▷   ▷   {
fecef9 ▷   ▷   ▷   ▷   result = PrintFileError(state,
513443 ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   "ERROR: Format too new for "
fc7968 ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   "this version of unmunge");
a01b7c ▷   ▷   ▷   ▷   goto error;
a4a3a6 ▷   ▷   ▷   }
71af5a
e6b210 ▷   ▷   ▷   p = strstr(p, " of ");
8f5a6d ▷   ▷   ▷   if (p == NULL)
47dda2 ▷   ▷   ▷   ▷   goto invalidHeader;
e8af5a
64f7b3 ▷   ▷   ▷   fileNameTail = p + 4;
c0d166 ▷   ▷   ▷   p = fileNameTail + strlen(fileNameTail);
6316ff ▷   ▷   ▷   if (p < fileNameTail + 3 ¦¦ p[-1] != '\n')
80dda2 ▷   ▷   ▷   ▷   goto invalidHeader;
bebb8c ▷   ▷   ▷   else
61acdd ▷   ▷   ▷   ▷   p[-1] = '\0';
6eaf5a
9caa92 ▷   ▷   ▷   if (state->out != NULL && state->pageCRC != state->seenPageCRC)
0df776 ▷   ▷   ▷   {
87cef9 ▷   ▷   ▷   ▷   result = PrintFileError(state,
54f1cd ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   "ERROR: Page CRC mismatch on page before");
f71b7c ▷   ▷   ▷   ▷   goto error;
cba3a6 ▷   ▷   ▷   }
5baf5a
929a4b ▷   ▷   ▷   if ((state->hdrFlags & HDR_FLAG_LASTPAGE) && state->out != NULL)
10f776 ▷   ▷   ▷   {
0d9d39 ▷   ▷   ▷   ▷   fclose(state->out);
8038bc ▷   ▷   ▷   ▷   state->out = NULL;
5ba3a6 ▷   ▷   ▷   }
e3af5a
3a8072 ▷   ▷   ▷   if (state->out != NULL)
e2f776 ▷   ▷   ▷   {
c1deba ▷   ▷   ▷   ▷   if (pageNumber != state->pageNumber + 1 ¦¦
1c0b4e ▷   ▷   ▷   ▷   ▷   ▷   fileNumber != state->fileNumber ¦¦
a2bf72 ▷   ▷   ▷   ▷   ▷   ▷   productNumber != state->productNumber ¦¦
```

```
0ac0fd ▷   ▷   ▷   ▷   ▷   ▷   tabWidth != state->tabWidth ¦¦
cc7b49 ▷   ▷   ▷   ▷   ▷   ▷   strcmp(fileNameTail, state->fileNameTail) != 0)
0872c3 ▷   ▷   ▷   ▷   {
a9f3b0 ▷   ▷   ▷   ▷   ▷   if (fileNumber == state->fileNumber &&
f852e6 ▷   ▷   ▷   ▷   ▷   ▷   ▷   pageNumber > state->pageNumber + 1)
874859 ▷   ▷   ▷   ▷   ▷   {
d41a41 ▷   ▷   ▷   ▷   ▷   ▷   (void)PrintFileError(state,
7035a ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   "ERROR: Missing pages of this file");
62d686 ▷   ▷   ▷   ▷   ▷   ▷   if (forcePartialFiles && !state->binaryMode)
008fec ▷   ▷   ▷   ▷   ▷   ▷   {
027341 ▷   ▷   ▷   ▷   ▷   ▷   ▷   fputs("\n\n@@@@@@ Missing pages here! @@@@@@\n\n",
e72f94 ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ‘‘state->out);
b7db3c ▷   ▷   ▷   ▷   ▷   ▷   }
957f8b ▷   ▷   ▷   ▷   ▷   ▷   else
678fec ▷   ▷   ▷   ▷   ▷   ▷   {
2dcf0a ▷   ▷   ▷   ▷   ▷   ▷   ▷   skipNextPage = 1;
1cc85a ▷   ▷   ▷   ▷   ▷   ▷   ▷   fclose(state->out);
68e69b ▷   ▷   ▷   ▷   ▷   ▷   ▷   state->out = NULL;
e63e60 ▷   ▷   ▷   ▷   ▷   ▷   ▷   remove(state->fileName);
24db3c ▷   ▷   ▷   ▷   ▷   ▷   }
1c1c89 ▷   ▷   ▷   ▷   ▷   }
d84bd6 ▷   ▷   ▷   ▷   ▷   else
e84859 ▷   ▷   ▷   ▷   ▷   {
451a41 ▷   ▷   ▷   ▷   ▷   ▷   (void)PrintFileError(state,
87974f ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   "ERROR: Missing pages of previous file");
86d686 ▷   ▷   ▷   ▷   ▷   ▷   if (forcePartialFiles && !state->binaryMode)
d68fec ▷   ▷   ▷   ▷   ▷   ▷   {
a77341 ▷   ▷   ▷   ▷   ▷   ▷   ▷   fputs("\n\n@@@@@@ Missing pages here! @@@@@@\n\n",
be2f94 ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ‘‘state->out);
da869e ▷   ▷   ▷   ▷   ▷   ▷   ▷   /* Make it non-fatal, though... */
f2c85a ▷   ▷   ▷   ▷   ▷   ▷   ▷   fclose(state->out);
43e69b ▷   ▷   ▷   ▷   ▷   ▷   ▷   state->out = NULL;
6edb3c ▷   ▷   ▷   ▷   ▷   ▷   }
7f7f8b ▷   ▷   ▷   ▷   ▷   ▷   else
0e8fec ▷   ▷   ▷   ▷   ▷   ▷   {
3ec85a ▷   ▷   ▷   ▷   ▷   ▷   ▷   fclose(state->out);
9be69b ▷   ▷   ▷   ▷   ▷   ▷   ▷   state->out = NULL;
593e60 ▷   ▷   ▷   ▷   ▷   ▷   ▷   remove(state->fileName);
26db3c ▷   ▷   ▷   ▷   ▷   ▷   }
841c89 ▷   ▷   ▷   ▷   ▷   }
f32613 ▷   ▷   ▷   ▷   }
1ba3a6 ▷   ▷   ▷   }
ec3fc9 ▷   ▷   ▷   if (state->out == NULL)
21f776 ▷   ▷   ▷   {
2155d5 ▷   ▷   ▷   ▷   if (pageNumber != 1 && !skipPage)
908c00 ▷   ▷   ▷   ▷   ▷   (void)PrintFileError(state,
6c16a6 ▷   ▷   ▷   ▷   ▷   ▷   ▷   ‘"ERROR: File doesn’t begin with page 1");
22af5a
80f7d0 ▷   ▷   ▷   ▷   state->binaryMode = (tabWidth == 0);
d9af5a
a62d94 ▷   ▷   ▷   ▷   if (pageNumber != 1 && (state->binaryMode
8bcd6f ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ¦¦ !forcePartialFiles))
af72c3 ▷   ▷   ▷   ▷   {
1aca21 ▷   ▷   ▷   ▷   ▷   skipNextPage = 1;
062613 ▷   ▷   ▷   ▷   }
aafa07 ▷   ▷   ▷   ▷   else
2972c3 ▷   ▷   ▷   ▷   {
22af45 ▷   ▷   ▷   ▷   ▷   /* TODO: Use global filelist to get pathname */
17c25c ▷   ▷   ▷   ▷   ▷   result = ReadManifest(state, fileNumber, fileNameTail,
fa6e0f ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ‘‘strlen(fileNameTail));
3eb9b8 ▷   ▷   ▷   ▷   ▷   if (result != 0)
d23dde ▷   ▷   ▷   ▷   ▷   ▷   goto error;
e5af5a
cba48a ▷   ▷   ▷   ▷   ▷   if (!forceOverwrite)
b24859 ▷   ▷   ▷   ▷   ▷   {
33a607 ▷   ▷   ▷   ▷   ▷   ▷   FILE *▷ file;
21af5a
769299 ▷   ▷   ▷   ▷   ▷   ▷   /* Make sure file doesn’t already exist */
84ee9f ▷   ▷   ▷   ▷   ▷   ▷   file = fopen(state->fileName, "r");
7ac7c5 ▷   ▷   ▷   ▷   ▷   ▷   if (file != NULL)
9e8fec ▷   ▷   ▷   ▷   ▷   ▷   {
3edb5d ▷   ▷   ▷   ▷   ▷   ▷   ▷   fclose(file);
```

```
--c6e5 ØØØ18d318c74ØØ2ØØØf Page 7 of unmunge.c

d7da9e ▷   ▷   ▷   ▷   ▷   ▷   ▷   fprintf(stderr, "ERROR: %s already exists\n",
8ec397 ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   state->fileName);
7c7f93 ▷   ▷   ▷   ▷   ▷   ▷   ▷   result = 1;
28f88b ▷   ▷   ▷   ▷   ▷   ▷   ▷   goto error;
3Ødb3c ▷   ▷   ▷   ▷   ▷   ▷   }
141c89 ▷   ▷   ▷   ▷   ▷   }
e3af5a
d9dØ71 ▷   ▷   ▷   ▷   ▷   state->out = fopen(state->fileName,
Ø62d32 ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ···state->binaryMode ? "wb" : "w");
5772fa ▷   ▷   ▷   ▷   ▷   if (state->out == NULL)
4266c8 ▷   ▷   ▷   ▷   ▷   ▷   goto errnoError;
d9af5a
ee892f ▷   ▷   ▷   ▷   ▷   if (pageNumber != 1)
Øc9e66 ▷   ▷   ▷   ▷   ▷   ▷   fputs("\n\n@@@@@@ Missing pages here! @@@@@@\n\n",
aabeØc ▷   ▷   ▷   ▷   ▷   ▷   ▷   ··state->out);
a62613 ▷   ▷   ▷   ▷   }
1ea3a6 ▷   ▷   ▷   }
4baf5a
e8b965 ▷   ▷   ▷   state->pageCRC = Ø;
cb479d ▷   ▷   ▷   state->seenPageCRC = seenPageCRC;
857936 ▷   ▷   ▷   state->hdrFlags = (word16)flags;
e9Ø8Ø6 ▷   ▷   ▷   state->pageNumber = pageNumber;
34d317 ▷   ▷   ▷   state->fileNumber = fileNumber;
39f9c2 ▷   ▷   ▷   state->productNumber = productNumber;
f3e34b ▷   ▷   ▷   state->tabWidth = tabWidth;
9ad7e7 ▷   ▷   ▷   skipPage = skipNextPage;
aØØfe4 ▷   ▷   }
ddb768 ▷   ▷   else if (!skipPage)
Ø85b34 ▷   ▷   {
da3fc9 ▷   ▷   ▷   if (state->out == NULL)
b5f776 ▷   ▷   ▷   {
Øf1ea3 ▷   ▷   ▷   ▷   result = PrintFileError(state, "ERROR: Missing header line");
161b7c ▷   ▷   ▷   ▷   goto error;
a4a3a6 ▷   ▷   ▷   }
deaf5a
caØbb5 ▷   ▷   ▷   /* Normal data line */
5cc493 ▷   ▷   ▷   state->pageCRC = CalculateCRC(fmt->pageCRC, state->pageCRC,
9f8744 ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ···(byte const *)lineData,
d69Øe3 ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ···length);
fa543e ▷   ▷   ▷   line[2] = '\Ø';
de71bc ▷   ▷   ▷   if (DecodeCheckDigits(fmt, line, NULL, fmt->runningCRCBits, &num)
6e311d ▷   ▷   ▷   ▷   ¦¦ RunningCRCFromPageCRC(fmt, state->pageCRC) != num)
8df776 ▷   ▷   ▷   {
6Ød578 ▷   ▷   ▷   ▷   result = PrintFileError(state, "ERROR: Running CRC failed");
ad1b7c ▷   ▷   ▷   ▷   goto error;
e6a3a6 ▷   ▷   ▷   }
92af5a
ØbØ99a ▷   ▷   ▷   if (state->binaryMode)
4Øf776 ▷   ▷   ▷   {
621ØØc ▷   ▷   ▷   ▷   length = DecodeLine(lineData, outbuf, length-1);
dbca8Ø ▷   ▷   ▷   ▷   if (length < Ø ¦¦ length > BYTES_PER_LINE) {
5a164b ▷   ▷   ▷   ▷   ▷   result = PrintFileError(state,
a7e761 ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   "ERROR: Corrupt radix-64 data");
f8b622 ▷   ▷   ▷   ▷   ▷   goto error;
9e2613 ▷   ▷   ▷   ▷   }
18Ø2b8 ▷   ▷   ▷   ▷   fwrite(outbuf, 1, length, state->out);
27a3a6 ▷   ▷   ▷   }
43bb8c ▷   ▷   ▷   else
84f776 ▷   ▷   ▷   {
28aØ9a ▷   ▷   ▷   ▷   p = lineData;
ead53d ▷   ▷   ▷   ▷   while (*p != '\Ø')
d572c3 ▷   ▷   ▷   ▷   {
2f1ea2 ▷   ▷   ▷   ▷   ▷   if (*p == TAB_CHAR)
f84859 ▷   ▷   ▷   ▷   ▷   {
e9aØcf ▷   ▷   ▷   ▷   ▷   ▷   p++;
ØbØf42 ▷   ▷   ▷   ▷   ▷   ▷   putc('\t', state->out);
8a8a27 ▷   ▷   ▷   ▷   ▷   ▷   while ((p - lineData) % state->tabWidth)
1a8fec ▷   ▷   ▷   ▷   ▷   ▷   {
3cb86e ▷   ▷   ▷   ▷   ▷   ▷   ▷   if (*p == '\n')
66cd67 ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   break;
9cb18a ▷   ▷   ▷   ▷   ▷   ▷   ▷   else if (*p == ' ')
1826f1 ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   p++;
```

```
8bc4dd ▷   ▷   ▷   ▷   ▷   ▷   ▷   else
91006b8 ▷   ▷   ▷   ▷   ▷   ▷   ▷   {
ee00b5 ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   result = PrintFileError(state,
cf82fb ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   "ERROR: Not enough spaces "
05d91a ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   "after a tab character");
a770ed ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   goto error;
c05268 ▷   ▷   ▷   ▷   ▷   ▷   ▷   }
9edb3c ▷   ▷   ▷   ▷   ▷   ▷   }
551c89 ▷   ▷   ▷   ▷   ▷   }
cf2256 ▷   ▷   ▷   ▷   ▷   else if (*p == FORMFEED_CHAR)
884859 ▷   ▷   ▷   ▷   ▷   {
bba0cf ▷   ▷   ▷   ▷   ▷   ▷   p++;
298fb3 ▷   ▷   ▷   ▷   ▷   ▷   if (*p != '\n')
ae8fec ▷   ▷   ▷   ▷   ▷   ▷   {
ae0057 ▷   ▷   ▷   ▷   ▷   ▷   ▷   result = PrintFileError(state,
9dfd7c ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   "ERROR: Formfeed character "
cac7f9 ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   "not at end of line");
14f88b ▷   ▷   ▷   ▷   ▷   ▷   ▷   goto error;
a8db3c ▷   ▷   ▷   ▷   ▷   ▷   }
797843 ▷   ▷   ▷   ▷   ▷   ▷   p++;▷   /* Skip newline */
0127bd ▷   ▷   ▷   ▷   ▷   ▷   putc('\f', state->out);
501c89 ▷   ▷   ▷   ▷   ▷   }
b0352e ▷   ▷   ▷   ▷   ▷   else if (*p == CONTIN_CHAR)
2c4859 ▷   ▷   ▷   ▷   ▷   {
1fa0cf ▷   ▷   ▷   ▷   ▷   ▷   p++;
9d8fb3 ▷   ▷   ▷   ▷   ▷   ▷   if (*p != '\n')
c08fec ▷   ▷   ▷   ▷   ▷   ▷   {
bb0057 ▷   ▷   ▷   ▷   ▷   ▷   ▷   result = PrintFileError(state,
a3538a ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   "ERROR: Continuation character "
14c7f9 ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   "not at end of line");
d2f88b ▷   ▷   ▷   ▷   ▷   ▷   ▷   goto error;
f2db3c ▷   ▷   ▷   ▷   ▷   ▷   }
937843 ▷   ▷   ▷   ▷   ▷   ▷   p++;▷   /* Skip newline */
ee1c89 ▷   ▷   ▷   ▷   ▷   }
e4ffb4 ▷   ▷   ▷   ▷   ▷   else if (*p == SPACE_CHAR)
244859 ▷   ▷   ▷   ▷   ▷   {
96c4c0 ▷   ▷   ▷   ▷   ▷   ▷   putc(' ', state->out);
59a0cf ▷   ▷   ▷   ▷   ▷   ▷   p++;
af1c89 ▷   ▷   ▷   ▷   ▷   }
fb4bd6 ▷   ▷   ▷   ▷   ▷   else
b54859 ▷   ▷   ▷   ▷   ▷   {
a09254 ▷   ▷   ▷   ▷   ▷   ▷   putc(*p, state->out);
68a0cf ▷   ▷   ▷   ▷   ▷   ▷   p++;
c91c89 ▷   ▷   ▷   ▷   ▷   }
a22613 ▷   ▷   ▷   ▷   }
8ea3a6 ▷   ▷   ▷   }
900fe4 ▷   ▷   }
f39371 ▷   }
48a6d0 ▷   if (state->out != NULL)
2fc7a1 ▷   {
4f62be ▷   ▷   if (!(state->hdrFlags & HDR_FLAG_LASTPAGE))
825b34 ▷   ▷   {
ea8347 ▷   ▷   ▷   result = PrintFileError(state, "ERROR: Missing pages");
24ed19 ▷   ▷   ▷   goto error;
3c0fe4 ▷   ▷   }
8a3d5d ▷   ▷   if (state->pageCRC != state->seenPageCRC)
185b34 ▷   ▷   {
875f61 ▷   ▷   ▷   result = PrintFileError(state,
54cd1f ▷   ▷   ▷   ▷   ▷   ▷   ▷   "ERROR: Page CRC failed on previous page");
cfed19 ▷   ▷   ▷   goto error;
200fe4 ▷   ▷   }
fa9371 ▷   }
a1af5a
858dfe ▷   /* Check for missing files at the end */
42edce ▷   result = ReadManifest(state, 0, NULL, 0);
42c48e ▷   goto done;
79af5a
e9d589 errnoError:
d816a9 ▷   result = errno;
31c66a ▷   goto printError;
b6af5a
327451 fileError:
```

```
22d1Ø9 ▷   result = ferror(state->file);
deaf5a
797374 printError:
443Ø89 ▷   fprintf(stderr, "ERROR: %s\n", strerror(result));
9Øaf5a
726Øff error:
66b38c done:
defa1f ▷   if (state != NULL)
1bc7a1 ▷   {
6aØb8e ▷   ▷   if (state->out != NULL)
6716c5 ▷   ▷   ▷   fclose(state->out);
f3Ø4dØ ▷   ▷   if (state->file != NULL)
3Øfc24 ▷   ▷   ▷   fclose(state->file);
cc4441 ▷   ▷   if (state->manifest != NULL)
3ff6fa ▷   ▷   ▷   fclose(state->manifest);
b7cb57 ▷   ▷   free(state);
359371 ▷   }
d55d73 ▷   return result;
78efe6 }
8faf5a
b21926 void UsageAndExit(int result)
18bb36 {
62c239 ▷   fprintf(stderr,
d847fb ▷   ▷   ▷   "Usage: unmunge [-fp] <file> [<manifest>]\n"
2d6aØ9 ▷   ▷   ▷   " ▪-f ▪Force overwrites of existing files\n"
45d1da ▷   ▷   ▷   " ▪-p ▪Force unmunge of partial files\n");
Ø6bcba ▷   exit(result);
97efe6 }
52af5a
c89eØ6 int main(int argc, char *argv[])
56bb36 {
3512eb ▷   int▷▷   result = Ø;
c61b78 ▷   int▷▷   forceOverwrite = Ø;
8Ø2be8 ▷   int▷▷   forcePartialFiles = Ø;
12e2Ø6 ▷   char *▷ fileName = NULL;
47Øacb ▷   char *▷ manifestFileName = NULL;
cd7be3 ▷   int▷▷   i, j;
85af5a
c966bd ▷   InitUtil();
7eaf5a
a366f8 ▷   for (i = 1; i < argc && argv[i][Ø] == '-'; i++)
edc7a1 ▷   {
c77b73 ▷   ▷   if (Ø == strcmp(argv[i], "--"))
7e5b34 ▷   ▷   {
cd81ec ▷   ▷   ▷   i++;
55cdcb ▷   ▷   ▷   break;
e9Øfe4 ▷   ▷   }
1ff864 ▷   ▷   for (j = 1; argv[i][j] != '\Ø'; j++)
275b34 ▷   ▷   {
d3dcd6 ▷   ▷   ▷   if (argv[i][j] == 'h')
d7c3ff ▷   ▷   ▷   ▷   UsageAndExit(Ø);
2eØØ5d ▷   ▷   ▷   else if (argv[i][j] == 'f')
7b6e59 ▷   ▷   ▷   ▷   forceOverwrite = 1;
2Ø8866 ▷   ▷   ▷   else if (argv[i][j] == 'p')
dba284 ▷   ▷   ▷   ▷   forcePartialFiles = 1;
1Øbb8c ▷   ▷   ▷   else
2bf776 ▷   ▷   ▷   {
13e685 ▷   ▷   ▷   ▷   fprintf(stderr, "ERROR: Unrecognized option -%c\n", argv[i][j]);
f5df44 ▷   ▷   ▷   ▷   UsageAndExit(1);
fba3a6 ▷   ▷   ▷   }
e1Øfe4 ▷   ▷   }
3e9371 ▷   }
37af5a
cØ498f ▷   if (i < argc)
8cØØ92 ▷   ▷   fileName = argv[i++];
23498f ▷   if (i < argc)
8cfe3e ▷   ▷   manifestFileName = argv[i++];
eØ6c14 ▷   if (fileName == NULL ¦¦ i < argc)
2b5Ø4f ▷   ▷   UsageAndExit(1);
95af5a
3f992c ▷   if ((result = UnMungeFile(fileName, manifestFileName,
4d86ab ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▪▪forceOverwrite, forcePartialFiles)) != Ø)
```

```
f7c7a1 ▷   {
4a654a ▷   ▷   /* If result > Ø, message should have already been printed */
15d72e ▷   ▷   if (result < Ø)
7cd1fe ▷   ▷   ▷   fprintf(stderr, "ERROR: %s\n", strerror(result));
Øc4bb3 ▷   ▷   exit(1);
Ø29371 ▷   }
f8af5a
4d3e6b ▷   return Ø;
91efe6 }
aaaf5a
df38e5 /*
a3e6c5 ‘* Local Variables:
e69b19 ‘* tab-width: 4
42e7a4 ‘* End:
Ø4b612 ‘* vi: ts=4 sw=4
336c42 ‘* vim: si
be495d ‘*/
c7af5a
```

```
--3793 ØØØ8ØØ1489d4ØØ2ØØ1Ø Page 1 of munge.c

bc38e5 /*
d5e662 ·* munge.c -- Program to convert a text file into "munged" form,
4fc8ee ·* ··········suitable for reconstruction from printed form. ·Tabs are
e7acdb ·* ··········made visible and checksums are added to each line and each
787f67 ·* ··········page to protect against transcription errors.
eØ775e ·*
Ø4a5ec ·* Copyright (C) 1997 Pretty Good Privacy, Inc.
6a775e ·*
dfbd99 ·* Designed by Colin Plumb, Mark H. Weaver, and Philip R. Zimmermann
af7659 ·* Written by Mark H. Weaver
e7775e ·*
459Ø28 ·* $Id: munge.c,v 1.32 1997/11/12 23:28:53 mhw Exp $
b9495d ·*/
ebaf5a
93feb2 #include <stdio.h>
Øf495d #include <errno.h>
c8324c #include <string.h>
4db1cb #include <ctype.h>
4fbea3 #include <stdlib.h>
62af5a
ff49Øa #include "util.h"
c3af5a
2338e5 /*
3c3e8e ·* The file is divided into pages, and the format of each page is
cc775e ·*
245Ø85 --f414 ØØØb2dc79af4ØØ1ØØØ2 Page 1 of munge.c
45af5a
9fdb26 bc38e5 /*
8fb899 4Øa838 ·* munge.c -- Program to convert a text file into munged form
Øac4d8 647222 ·*
fbc7a9 193f28 ·* Copyright (C) 1997 Pretty Good Privacy, Inc.
5ee13c 827222 ·*
96723a 699Ø25 ·* Designed by Colin Plumb, Mark H. Weaver, and Philip R. Zimmermann
237Ø55 ØdØ5Øc ·* Written by Mark H. Weaver
62775e ·*
Ø2323b ·* Where the first 2 columns are the high 8 bits (in hex) of a running
452e82 ·* CRC-32 of the page (the string "--", unlikely to be confused with
34345f ·* any digits, indicates a page header line) and the next 4 columns
3e7357 ·* are a CRC-16 of the rest of the line. ·Then a space (not counted in
8bc9ab ·* the CRC), and the line of text. ·Tabs are printed as the currency
d8aØ4a ·* symbol (ISO Latin 1 character 164) followed by the appropriate number
3d193Ø ·* of spaces, and any form feeds are printed as a yen symbol (Latin 1 165).
bcb41d ·* The CRC is computed on the transformed line, including the trailing
f68916 ·* newline. ·No trailing whitespace is permitted.
cd775e ·*
bfbØ9c ·* The header line contains a (hex) number of the form Øffccccccctpppnnnn,
92Ø6c8 ·* where the digit Ø is a version number, ff are flags, ccccccc is the CRC-32
f8fØ55 ·* of the page, t is the tab size (usually 4 or 8; Ø for binary files that
1df99e ·* are sent in radix-64), ppp is the product number (usually 1, different
53dbd7 ·* for different books), and nnnn is the file number (sequential from 1).
51775e ·*
86d19a ·* This is followed by " Page %u of " and the file name.
33495d ·*/
c7af5a
Ø1139Ø typedef struct MungeState
e2bb36 {
8935f4 ▷   EncodeFormat const *▷   fmt;
bfd912 ▷   EncodeFormat const *▷   hFmt;
c37ffa ▷   int▷▷    ▷    ▷   binaryMode, tabWidth;
Ø2c575 ▷   long▷    ▷    ▷   origLineNumber;
Ø17744 ▷   long▷    ▷    ▷   productNumber, fileNumber, pageNumber, lineNumber;
f7aedd ▷   unsigned long▷  fileOffset;
1e2ec2 ▷   CRC▷▷    ▷    ▷   pageCRC;
913a63 ▷   char const *▷   fileName;
a628a3 ▷   char const *▷   fileNameTail;
df2Ød4 ▷   char *▷ ▷    ▷   pageBuffer;▷/* Buffer large enough to hold one page */
b949f9 ▷   char *▷ ▷    ▷   pagePos;▷   /* Current position in pageBuffer */
989ad5 ▷   word16▷ ▷    ▷   hdrFlags;
38237d ▷   FILE *▷ ▷    ▷   file;
4f4Ø43 ▷   FILE *▷ ▷    ▷   out;
64e329 } MungeState;
8Øaf5a
```

```
eØaf5a
6b596a void ChecksumLine(EncodeFormat const *fmt, char const *line, size_t length,
a6237d ▷   ▷   ▷   ▷   ▷▷char *prefix, CRC *pageCRC)
a8bb36 {
e6cfbf ▷   CRC▷▷   ▷   lineCRC;
9d2Ø15 ▷   CRC▷▷   ▷   runCRCPart = Ø;
35af5a
638844 ▷   lineCRC = CalculateCRC(fmt->lineCRC, Ø, (byte const *)line, length);
f63683 ▷   if (pageCRC != NULL)
b9c7a1 ▷   {
55a5c8 ▷   ▷   *pageCRC = CalculateCRC(fmt->pageCRC, *pageCRC,
df35d7 ▷   ▷   ▷   ▷   ▷   ▷   ▷   (byte const *)line, length);
Øa462b ▷   ▷   runCRCPart = RunningCRCFromPageCRC(fmt, *pageCRC);
9c9371 ▷   }
b2af5a
d42786 ▷   prefix += EncodeCheckDigits(fmt, runCRCPart, fmt->runningCRCBits, prefix);
baa45f ▷   prefix += EncodeCheckDigits(fmt, lineCRC, fmt->lineCRC->bits, prefix);
cbaf5a
2f4b71 ▷   *prefix++ = ' ';▷   /* Write a space over the null byte */
dcefe6 }
e9af5a
d6533a /* Returns 1 for convenience */
24310b int PrintFileError(MungeState *state, char const *message)
45bb36 {
9c890a ▷   fprintf(stderr, "%s in %s %s %lu\n", message, state->fileName,
1Øf2d1 ▷   ▷   ▷   state->binaryMode ? "offset" : "line",
825129 ▷   ▷   ▷   state->binaryMode ? state->fileOffset : state->origLineNumber);
1b64b7 ▷   return 1;
3Øefe6 }
76af5a
2d6ba3 int MungeLine(MungeState *state, char *buffer, int length,
bdc6Øb ▷   ▷   ▷   ▷▷char *line, int *bufferUsed)
bcbb36 {
c9ac66 ▷   int▷▷   i = Ø, j = Ø, jOld = Ø;
cab93e ▷   char▷   ch;
cdaf5a
c2bb55 ▷   for (i = Ø; i < length && j < LINE_LENGTH; i++)
14c7a1 ▷   {
eØ9f74 ▷   ▷   jOld = j;
e37479 ▷   ▷   ch = buffer[i];
fd2fa3 ▷   ▷   if (ch == '\t')
d95b34 ▷   ▷   {
9b890b ▷   ▷   ▷   line[j++] = TAB_CHAR;
5265ba ▷   ▷   ▷   if (state->tabWidth < 1)
c79386 ▷   ▷   ▷   ▷   return PrintFileError(state,
1ac5bc ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷▷"ERROR: Tab found in radix64 stream");
8cbb8c ▷   ▷   ▷   else
ca7c85 ▷   ▷   ▷   ▷   while (j % state->tabWidth && j < LINE_LENGTH)
c98a1a ▷   ▷   ▷   ▷   ▷   line[j++] = TAB_PAD_CHAR;
9dØfe4 ▷   ▷   }
23d22e ▷   ▷   else if (ch == '\n')
af5b34 ▷   ▷   {
5e7e1b ▷   ▷   ▷   if (i + 1 < length)
ed9386 ▷   ▷   ▷   ▷   return PrintFileError(state,
e26baa ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   "UNEXPECTED ERROR: fgets read past newline!?");
5fcdcb ▷   ▷   ▷   break;
8eØfe4 ▷   ▷   }
bd37f6 ▷   ▷   else if (ch == '\f')
555b34 ▷   ▷   {
7bcdcb ▷   ▷   ▷   break;
c3Øfe4 ▷   ▷   }
1cb49Ø ▷   ▷   else if (ch == ' ' && (j <= Ø ¦¦ line[j-1] == ' ' ¦¦
11a379 ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷▷▷line[j-1] == SPACE_CHAR ¦¦
3fc3Øa ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷▷▷i+1 >= length ¦¦ buffer[i+1] == '\n'))
7a5b34 ▷   ▷   {
9931bb ▷   ▷   ▷   line[j++] = SPACE_CHAR;
6e3b9Ø ▷   ▷   }▷
9eebee ▷   ▷   else if (ch >= ' ' && ch <= '~')
245c41 ▷   ▷   ▷   line[j++] = ch;
12Øe1d ▷   ▷   else
968ef7 ▷   ▷   ▷   return PrintFileError(state, "ERROR: Non-ASCII char");
Ø89371 ▷   }
```

```
e0af5a
df1ded ▷   if (i < length && buffer[i] == '\n')
7cc7a1 ▷   {
bb347d ▷   ▷   i++;
e887d1 ▷   ▷   state->origLineNumber++;
5Ø9371 ▷   }
6825be ▷   else if (i < length && buffer[i] == '\f' && j < LINE_LENGTH)
f3c7a1 ▷   {
46347d ▷   ▷   i++;
b87d31 ▷   ▷   line[j++] = FORMFEED_CHAR;
Øc9371 ▷   }
fcØe24 ▷   else
fec7a1 ▷   {
fØda77 ▷   ▷   /* If there's no newline, we need to add the continuation marker */
aØ7ab1 ▷   ▷   if (i > Ø && j >= LINE_LENGTH)
3f5b34 ▷   ▷   {
b8dcfc ▷   ▷   ▷   /* Remove the last character if we're out of room */
fØ1caf ▷   ▷   ▷   i--;
5f8ce6 ▷   ▷   ▷   j = jOld;
93Øfe4 ▷   ▷   }
Ø36673 ▷   ▷   line[j++] = CONTIN_CHAR;
Ø29371 ▷   }
81af5a
e6d777 ▷   /* Strip trailing spaces */
18151b ▷   while (j > Ø && isspace((unsigned char)line[j - 1]))
c6b4f2 ▷   ▷   j--;
Ø2af5a
4a5483 ▷   if (j > LINE_LENGTH)▷   /* This should never happen */
366e7f ▷   ▷   return PrintFileError(state, "ERROR: Internal error, line too long");
cdaf5a
c59ad7 ▷   /* Add trailing newline and NULL */
64d85f ▷   line[j++] = '\n';
22a3Øb ▷   line[j++] = '\Ø';
e5af5a
Øa55be ▷   /* Return number of chars used from buffer */
cb69e5 ▷   *bufferUsed = i;
6eaf5a
d23e6b ▷   return Ø;
26efe6 }
6aaf5a
662978 static void
ce57e8 Encode3(byte const src[3], char dest[4])
c6bb36 {
fØØfed ▷   dest[Ø] = radix64Digits[ ‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘(src[Ø]>>2 & Øx3f)];
be9e38 ▷   dest[1] = radix64Digits[(src[Ø]<<4 & Øx3Ø) ¦ (src[1]>>4 & Øx Øf)];
c7d36b ▷   dest[2] = radix64Digits[(src[1]<<2 & Øx3c) ¦ (src[2]>>6 & Øx Ø3)];
2Øf35e ▷   dest[3] = radix64Digits[(src[2] ‘‘‘& Øx3f)];
cfefe6 }
3eaf5a
4fbeb7 static int
Ø4ee58 EncodeLine(byte const *src, int srcLen, char *dest)
bØbb36 {
f64365 ▷   char *▷ destp = dest;
5Øf3c2 ▷   byte▷   tempSrc[3];
41af5a
4f8feb ▷   for (; srcLen >= 3; srcLen -= 3)
Øbc7a1 ▷   {
e399a7 ▷   ▷   Encode3(src, destp);
eaf225 ▷   ▷   src += 3; destp += 4;
ba9371 ▷   }
Ø8af5a
92f7f1 ▷   if (srcLen > Ø)
b3c7a1 ▷   {
2Ø34bØ ▷   ▷   memset(tempSrc, Ø, sizeof(tempSrc));
9122Øc ▷   ▷   memcpy(tempSrc, src, srcLen);
c699a7 ▷   ▷   Encode3(src, destp);
de844b ▷   ▷   src += 3; destp += 4; srcLen -= 3;
933792 ▷   ▷   while (srcLen < Ø)
66f8be ▷   ▷   ▷   destp[srcLen++] = RADIX64_END_CHAR;
559371 ▷   }
2aaf5a
bedd32 ▷   return destp - dest;
```

```
7aefe6 }
46af5a
88beb7 static int
9eb9ab MungeBinaryLine(MungeState *state, byte const *buffer, int length, char *line)
a3bb36 {
839bd9 ▷   char▷   binLine[128];
a7c54Ø ▷   int▷▷   binLength;▷ ▷   ▷   /* Destination length */
997fc2 ▷   int▷▷   used;
95af5a
5dØ93e ▷   binLength = EncodeLine(buffer, length, binLine);
Øaaf5a
69Øed4 ▷   /* Append newline */
5863a4 ▷   binLine[binLength++] = '\n';
d28645 ▷   binLine[binLength] = '\Ø';
6baf5a
af575Ø ▷   return MungeLine(state, binLine, binLength, line, &used);
d1efe6 }
e6af5a
95cad4 int MaybePageBreak(MungeState *state)
48bb36 {
dafØ17 ▷   EncodeFormat const *▷   fmt = state->fmt;
4Ø49ae ▷   EncodeFormat const *▷   hFmt = state->hFmt;
c2af5a
9c5e84 ▷   if (state->lineNumber >= LINES␣PER␣PAGE)
93c7a1 ▷   {
c49Ø7b ▷   ▷   char▷   line[512];
3e6aØ2 ▷   ▷   char *▷ lineData▷   = line + PREFIX␣LENGTH;
283e15 ▷   ▷   char *▷ p▷  ▷   ▷   = lineData;
3a1b52 ▷   ▷
2eca8Ø ▷   ▷   p += EncodeCheckDigits(hFmt, Ø, HDR␣VERSION␣BITS, p);
b5986d ▷   ▷   p += EncodeCheckDigits(hFmt, state->hdrFlags, HDR␣FLAG␣BITS, p);
424e27 ▷   ▷   p += EncodeCheckDigits(hFmt, state->pageCRC, fmt->pageCRC->bits, p);
2994a9 ▷   ▷   p += EncodeCheckDigits(hFmt, state->tabWidth, HDR␣TABWIDTH␣BITS, p);
55c3a6 ▷   ▷   p += EncodeCheckDigits(hFmt, state->productNumber, HDR␣PRODNUM␣BITS, p);
c1772Ø ▷   ▷   p += EncodeCheckDigits(hFmt, state->fileNumber, HDR␣FILENUM␣BITS, p);
82af5a
d8Øc2d ▷   ▷   sprintf(p, " Page %ld of %s\n", state->pageNumber + 1,
a13dc5 ▷   ▷   ▷   ▷   state->fileNameTail);
b4af5a
cdd23c ▷   ▷   if (strlen(lineData) > LINE␣LENGTH + 1)
6c5b34 ▷   ▷   {
44ddd9 ▷   ▷   ▷   PrintFileError(state, "ERROR: Header line too long");
3a5ØeØ ▷   ▷   ▷   fprintf(stderr, "> %s", lineData);
523fd5 ▷   ▷   ▷   return -1;
88Øfe4 ▷   ▷   }
dcaf5a
6bdc21 ▷   ▷   /* Compute checksums and prefix them to line */
7b2647 ▷   ▷   ChecksumLine(fmt, lineData, strlen(lineData), line, NULL);
15af5a
bØØa55 ▷   ▷   fprintf(state->out, "%c%c%s\n%s\f", HDR␣PREFIX␣CHAR,
5Ø8dØc ▷   ▷   ▷   ▷   fmt->headerTypeChar, line + 2, state->pageBuffer);
3faf5a
b4969f ▷   ▷   state->pageNumber++;
69db32 ▷   ▷   state->lineNumber = Ø;
b5143b ▷   ▷   state->pageCRC = Ø;
1aeda6 ▷   ▷   state->pagePos = state->pageBuffer;▷▷   /* Clear page buffer */
829371 ▷   }
2f3e6b ▷   return Ø;
86efe6 }
46af5a
9e38e5 /*
e6b468 ·* Search for Emacs "tab-width: " maker in file.
b86ca4 ·* Emacs is stricter about the format, but this will do.
46495d ·*/
9e3be5 int FindTabWidth(MungeState *state)
79bb36 {
6995b5 ▷   char const * const▷ tabWidthMarker = " tab-width: ";
e6cØdØ ▷   char▷   ▷   ▷   ▷   buffer[512];
Ød1bØa ▷   char *▷ ▷   ▷   ▷   p;
4cb442 ▷   int▷▷   ▷   ▷   ▷   length;
4a33bc ▷   int▷▷   ▷   ▷   ▷   tabWidth = Ø;
Ø3af5a
```

```
1d3ØØd ▷   fseek(state->file, -(sizeof(buffer) - 1), SEEK␣END);
Øf42ac ▷   length = fread(buffer, 1, sizeof(buffer) - 1, state->file);
696412 ▷   buffer[length] = '\Ø';
27a4ØØ ▷   p = strstr(buffer, tabWidthMarker);
6a3cde ▷   if (p != NULL)
b5c7a1 ▷   {
7f95Ø4 ▷   ▷   p += strlen(tabWidthMarker);
8ea56d ▷   ▷   while (*p != '\Ø' && *p != '\n' && isspace(*p))
2464c8 ▷   ▷   ▷   p++;
bb6233 ▷   ▷   tabWidth = strtol(p, &p, 1Ø);
34a56d ▷   ▷   while (*p != '\Ø' && *p != '\n' && isspace(*p))
ef64c8 ▷   ▷   ▷   p++;
e94c54 ▷   ▷   if (*p != '\n' ¦¦ tabWidth < 2)
6da16c ▷   ▷   ▷   tabWidth = Ø;
4167b8 ▷   ▷   else if (tabWidth > 16)
8c72f7 ▷   ▷   ▷   fprintf(stderr, "WARNING: Weird tab-width (%d), %s\n",
7aee6c ▷   ▷   ▷   ▷   ▷   ▷   ▷   tabWidth, state->fileName);
759371 ▷   }
234328 ▷   return tabWidth;
17efe6 }
3eaf5a
6c38e5 /*
89Ød91 ·* Open the given source file and send the munged output to the
c2Ø772 ·* FILE *, with the given options.
f8495d ·*/
5c9cØ7 int MungeFile(char const *fileName, FILE *out, EncodeFormat const *fmt,
13348e ▷   ▷   ▷   ··int binaryMode, int defaultTabWidth,
29252c ▷   ▷   ▷   ··long productNumber, long fileNumber)
91bb36 {
58a442 ▷   MungeState *▷   state;
8857bd ▷   int▷▷   ▷   ▷   length, used;
a8e9c9 ▷   char▷   ▷   ▷   line[PREFIX␣LENGTH + LINE␣LENGTH + 1Ø];
c9823f ▷   char *▷ ▷   ▷   lineData = line + PREFIX␣LENGTH;
Ø1Ø6bb ▷   char▷   ▷   ▷   buffer[128];
4acfe7 ▷   int▷▷   ▷   ▷   result = Ø;
fcaf5a
3e1e8b ▷   state = (MungeState *)calloc(1, sizeof(*state));
d7123b ▷   state->fmt = fmt;
9ab46d ▷   state->hFmt = &hexFormat;
fec93a ▷   state->origLineNumber = 1;
47Ø4aa ▷   state->fileName = fileName;
d9e25e ▷   state->pageCRC = Ø;
738f39 ▷   state->productNumber = productNumber;
849e24 ▷   state->fileNumber = fileNumber;
Ødd6cf ▷   state->pageNumber = Ø;
79cf1a ▷   state->lineNumber = Ø;
448b7c ▷   state->fileOffset = Ø;
Ø7bd37 ▷   state->binaryMode = binaryMode;
746634 ▷   state->pageBuffer = malloc(PAGE␣BUFFER␣SIZE);
bb5cØd ▷   state->pageBuffer[Ø] = '\Ø';
Ø65ba8 ▷   state->pagePos = state->pageBuffer;
8de846 ▷   state->hdrFlags = Ø;
4e3766 ▷   state->out = out;
35af5a
65ae1f ▷   state->fileNameTail = strrchr(state->fileName, '/');
c65fa8 ▷   if (state->fileNameTail == NULL)
d911d1 ▷   ▷   state->fileNameTail = state->fileName;
24Øe24 ▷   else
Ø2eØd3 ▷   ▷   state->fileNameTail++;
9daf5a
8Ø8f87 ▷   state->file = fopen(state->fileName, binaryMode ? "rb" : "r");
7aØØ3d ▷   if (state->file == NULL)
Ø9c7a1 ▷   {
9d6c5d ▷   ▷   result = errno;
87dd44 ▷   ▷   fprintf(stderr, "ERROR opening %s: %s\n",
cee1df ▷   ▷   ▷   ▷   state->fileName, strerror(result));
bØ97ed ▷   ▷   goto error;
789371 ▷   }
7d67c5 ▷
5Ø91cd ▷   if (state->binaryMode)
13c7a1 ▷   {
b8c843 ▷   ▷   state->tabWidth = Ø;
```

```
a19371 ▷   }
Øe Øe24 ▷   else
96c7a1 ▷   {
81Øa44 ▷   ▷   state->tabWidth = FindTabWidth(state);
425948 ▷   ▷   if (state->tabWidth == Ø)
66Ø8dd ▷   ▷   ▷   state->tabWidth = defaultTabWidth;
Ø5Ø8e2 ▷   ▷   rewind(state->file);
b99371 ▷   }
1eaf5a
b21978 ▷   while (!feof(state->file))
6bc7a1 ▷   {
3b85e5 ▷   ▷   if (state->binaryMode)
Ø45b34 ▷   ▷   {
3988e3 ▷   ▷   ▷   length = fread(buffer, 1, BYTES_PER_LINE, state->file);
df5257 ▷   ▷   ▷   if (length < 1)
eef776 ▷   ▷   ▷   {
49Ø52b ▷   ▷   ▷   ▷   if (feof(state->file))
2214c3 ▷   ▷   ▷   ▷   ▷   break;
46c87f ▷   ▷   ▷   ▷   goto fileError;
bba3a6 ▷   ▷   ▷   }
1b96a9 ▷   ▷   ▷   if ((result = MaybePageBreak(state)))
f81b7c ▷   ▷   ▷   ▷   goto error;
9eec4d ▷   ▷   ▷   if ((result = MungeBinaryLine(state, buffer, length, lineData)))
a91b7c ▷   ▷   ▷   ▷   goto error;
7b7b53 ▷   ▷   ▷   state->fileOffset += length;
ØØØfe4 ▷   ▷   }
33Øe1d ▷   ▷   else
d15b34 ▷   ▷   {
556ccb ▷   ▷   ▷   if (fgets(buffer, sizeof(buffer), state->file) == NULL)
2Øf776 ▷   ▷   ▷   {
7bØ52b ▷   ▷   ▷   ▷   if (feof(state->file))
4714c3 ▷   ▷   ▷   ▷   ▷   break;
ØØc87f ▷   ▷   ▷   ▷   goto fileError;
bØa3a6 ▷   ▷   ▷   }
d6d58a ▷   ▷   ▷   length = strlen(buffer);
8b96a9 ▷   ▷   ▷   if ((result = MaybePageBreak(state)))
8e1b7c ▷   ▷   ▷   ▷   goto error;
c23d54 ▷   ▷   ▷   if ((result = MungeLine(state, buffer, length, lineData, &used)))
aa1b7c ▷   ▷   ▷   ▷   goto error;
f6af5a
5c41d1 ▷   ▷   ▷   if (used < length)
713df4 ▷   ▷   ▷   ▷   if (fseek(state->file, used - length, SEEK_CUR))
2Ø4383 ▷   ▷   ▷   ▷   ▷   goto fileError;
8aØfe4 ▷   ▷   }
c8af5a
Ø77627 ▷   ▷   /* Compute checksums and prefix them to the line */
589fbØ ▷   ▷   ChecksumLine(fmt, lineData, strlen(lineData), line, &state->pageCRC);
bbaf5a
9a9562 ▷   ▷   strcpy(state->pagePos, line);
bdcØfb ▷   ▷   length = strlen(state->pagePos);
77e239 ▷   ▷   /* Suppress trailing whitespace on blank lines */
2f811a ▷   ▷   if (length == PREFIX_LENGTH+1 && state->pagePos[length-1] == '\n') {
424763 ▷   ▷   ▷   state->pagePos[--length-1] = '\n';
59f363 ▷   ▷   ▷   state->pagePos[length] = '\Ø';
e2Øfe4 ▷   ▷   }
c98d8d ▷   ▷   state->pagePos += length;
a2af5a
5cfe43 ▷   ▷   state->lineNumber++;
989371 ▷   }
81af5a
ca16ea ▷   if (state->lineNumber > Ø)
4dc7a1 ▷   {
cØ3987 ▷   ▷   /* Force a final page break */
e18Øe7 ▷   ▷   state->lineNumber = LINES_PER_PAGE;
e9112e ▷   ▷   state->hdrFlags ¦= HDR_FLAG_LASTPAGE;
5ad997 ▷   ▷   if ((result = MaybePageBreak(state)))
51ed19 ▷   ▷   ▷   goto error;
429371 ▷   }
dØaf5a
9fdd5a ▷   result = Ø;
5ec48e ▷   goto done;
2baf5a
```

```
957451 fileError:
41d1Ø9 ▷   result = ferror(state->file);
73af5a
616Øff error:
dcb38c done:
ebfa1f ▷   if (state != NULL)
c5c7a1 ▷   {
2ØØ4dØ ▷   ▷   if (state->file != NULL)
f5fc24 ▷   ▷   ▷   fclose(state->file);
92cb57 ▷   ▷   free(state);
3c9371 ▷   }
1Ø5d73 ▷   return result;
17efe6 }
f6af5a
979eØ6 int main(int argc, char *argv[])
a9bb36 {
f912eb ▷   int▷▷   result = Ø;
797be3 ▷   int▷▷   i, j;
d55396 ▷   int▷▷   defaultTabWidth = 4;
82cfc2 ▷   int▷▷   binaryMode = Ø;
b7d58b ▷   long▷   productNumber = 1;
Ø9183a ▷   long▷   fileNumber = 1;
dØ4173 ▷   char *▷ endOfNumber;
13d86f ▷   EncodeFormat const *▷   fmt = NULL;
d9af5a
3b66bd ▷   InitUtil();
6daf5a
b166f8 ▷   for (i = 1; i < argc && argv[i][Ø] == '-'; i++)
edc7a1 ▷   {
be7b73 ▷   ▷   if (Ø == strcmp(argv[i], "--"))
6e5b34 ▷   ▷   {
4881ec ▷   ▷   ▷   i++;
3Øcdcb ▷   ▷   ▷   break;
84Øfe4 ▷   ▷   }
64f864 ▷   ▷   for (j = 1; argv[i][j] != '\Ø'; j++)
dc5b34 ▷   ▷   {
92c73c ▷   ▷   ▷   if (isdigit(argv[i][j]))
84f776 ▷   ▷   ▷   {
4aebac ▷   ▷   ▷   ▷   defaultTabWidth = argv[i][j] - 'Ø';
4cff63 ▷   ▷   ▷   ▷   if (defaultTabWidth < 2 ¦¦ defaultTabWidth > 9)
7b9daa ▷   ▷   ▷   ▷   ▷   fprintf(stderr, "WARNING: Weird default tab-width (%d)\n",
f5faØ8 ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   defaultTabWidth);
d9a3a6 ▷   ▷   ▷   }
7472b1 ▷   ▷   ▷   else if (argv[i][j] == 'b')
e5f776 ▷   ▷   ▷   {
b389ab ▷   ▷   ▷   ▷   binaryMode = 1;
4da3a6 ▷   ▷   ▷   }
4a8fØe ▷   ▷   ▷   else if (argv[i][j] == 'F')
63f776 ▷   ▷   ▷   {
fa5d43 ▷   ▷   ▷   ▷   fmt = FindFormat(argv[i][j+1]);
77e1b2 ▷   ▷   ▷   ▷   if (!fmt ¦¦ argv[i][j+2] != '\Ø')
c572c3 ▷   ▷   ▷   ▷   {
fbØ9e7 ▷   ▷   ▷   ▷   ▷   fprintf(stderr, "ERROR: Invalid format char\n");
dc8fb4 ▷   ▷   ▷   ▷   ▷   exit(1);
fe2613 ▷   ▷   ▷   ▷   }
f3f622 ▷   ▷   ▷   ▷   break;
eca3a6 ▷   ▷   ▷   }
7d8866 ▷   ▷   ▷   else if (argv[i][j] == 'p')
fbf776 ▷   ▷   ▷   {
a1d9fa ▷   ▷   ▷   ▷   productNumber = strtol(&argv[i][j+1], &endOfNumber, 1Ø);
12154a ▷   ▷   ▷   ▷   if (*endOfNumber != '\Ø')
5a72c3 ▷   ▷   ▷   ▷   {
de85ca ▷   ▷   ▷   ▷   ▷   fprintf(stderr, "ERROR: Invalid product number\n");
3c8fb4 ▷   ▷   ▷   ▷   ▷   exit(1);
792613 ▷   ▷   ▷   ▷   }
25f622 ▷   ▷   ▷   ▷   break;
Øaa3a6 ▷   ▷   ▷   }
a8ØØ5d ▷   ▷   ▷   else if (argv[i][j] == 'f')
b9f776 ▷   ▷   ▷   {
Ø26de3 ▷   ▷   ▷   ▷   fileNumber = strtol(&argv[i][j+1], &endOfNumber, 1Ø);
a8154a ▷   ▷   ▷   ▷   if (*endOfNumber != '\Ø')
a172c3 ▷   ▷   ▷   ▷   {
```

```
8e2e4e ▷   ▷   ▷   ▷   ▷   fprintf(stderr, "ERROR: Invalid file number\n");
318fb4 ▷   ▷   ▷   ▷   ▷   exit(1);
9e2613 ▷   ▷   ▷   ▷   }
71f622 ▷   ▷   ▷   ▷   break;
46a3a6 ▷   ▷   ▷   }
57bb8c ▷   ▷   ▷   else
Øef776 ▷   ▷   ▷   {
61e685 ▷   ▷   ▷   ▷   fprintf(stderr, "ERROR: Unrecognized option -%c\n", argv[i][j]);
4fbbe9 ▷   ▷   ▷   ▷   exit(1);
17a3a6 ▷   ▷   ▷   }
b1Øfe4 ▷   ▷   }
c99371 ▷   }
721b9b ▷   if (!fmt)
5325Ø4 ▷   ▷   fmt = binaryMode ? &radix64Format : &hexFormat;
2aaf5a
f6527c ▷   for (; i < argc; i++)
25c7a1 ▷   {
4e2da1 ▷   ▷   if ((result = MungeFile(argv[i], stdout, fmt, binaryMode,
b73c3a ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   defaultTabWidth, productNumber,
325Øa4 ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   fileNumber)) != Ø)
745b34 ▷   ▷   {
1673fc ▷   ▷   ▷   /* If result > Ø, message should have already been printed */
15214b ▷   ▷   ▷   if (result < Ø)
5e6de7 ▷   ▷   ▷   ▷   fprintf(stderr, "ERROR: %s\n", strerror(result));
8dØa38 ▷   ▷   ▷   exit(1);
dØØfe4 ▷   ▷   }
ed4Ød5 ▷   ▷   fileNumber++;
5Ø9371 ▷   }
e767c5 ▷
d33e6b ▷   return Ø;
73efe6 }
c4af5a
f538e5 /*
7ce6c5 ·* Local Variables:
bf9b19 ·* tab-width: 4
6de7a4 ·* End:
b3b612 ·* vi: ts=4 sw=4
176c42 ·* vim: si
Ø7495d ·*/
```

```
--e4c5 ØØ1a2ea82a88ØØ2ØØ11 Page 1 of yapp.doc

13a533 YAPP is a simple macro preprocessor designed to do minor tweaking to
842adf another program's inputs.
8baf5a
ØØcfb6 In its input, anything of the form ${foo} is expanded with the variable
bØ46c3 named foo. ‣It is an error if ${foo} is not defined.
513117 If you need to escape a dollar sign for some reason, the variable
Ø1b9Øe with the empty string name , ${}, has the value "$".
13af5a
818c7Ø The result of macro expansion is *not* re-expanded. ‣Expansion is done only
cc2125 when definitions are made.
Ø6af5a
6d977d After variable expansion, lines are checked to see if they are control lines.
ØØ897c Control lines begin with ## (after optional leading whitespace) ‣All such lines
8e5d52 ‣are deleted and
dc1369 do not appear in the output. ‣### is a comment. ‣Other options
3311bc are:
3daf5a
4f6c2e ##set variable=value
aeaf5a
e13713 value may have one of the following forms:
e6923c token: ‣Trailing whitespace is stripped. ‣The token may not contain
ab411c any whitespace. ‣Use quotes if it's complicated.
5e88df "string": ‣The string may have embedded quotes, and whitespace after
b47521 ▷       the closing quote.
beac2b <<"DELIM": ‣This is a here-document, and the value is all of the following
7Ø75db lines up until, but not including, the newline that precedes a line
91694f that consists soley of DELIM, for any DELIM string.
75f189 The Delim must be in quotes. ‣You have two options:
289b94 "DELIM": Expand macros in the body of the here-document.
a6a481 'DELIM': Do not expand macros in the here-document.
82af5a
db9d79 ##include "filename": Insert the named file in place of the current line.
bbaf5a
833253 ##if num == num
6d36ØØ ##if num != num
1d5c38 ##if num < num
42546e ##if num > num
Øfad86 ##if num <= num
d99a3d ##if num >= num
7be216 ##if token eq token
32Ø67d ##if token ne token
cd2d35 ##ifdef symbol
e692Øa ##ifndef symbol
391Øcc ##else
784874 ##endif
18799b You can figure this one out. ‣Macros in between are expanded as usual
bcde1d (so the ##else or ##endif may be in a macro expansion), but the result
7af9af is ignored. ‣String comparison is allowed only between simple words.
a259a8 #ifdef symbol is true if ${symbol} is defined.
```

```
Øb326b #!/usr/bin/env perl
44a6Ø1 #
1af8b2 # Yet another preprocessor
39a6Ø1 #
5d76de # $Id: yapp,v 1.5 1997/1Ø/24 Ø7:51:Ø5 mhw Exp $
Ø9a6Ø1 #
27af5a
e2a771 %vars = ('' => '$');
b19c1a @incPath = (".");
dcaf5a
e32974 sub Error
8ebb36 {
2cf918 ▷   print STDERR $_[Ø], "\n";
81fe22 ▷   exit(1);
aaefe6 }
87af5a
a1Ø22f sub VarSubst
89bb36 {
bef38d ▷   my ($varName, $undefOkay) = @_;
e4af5a
8Ø68cØ ▷   if (defined($vars{$varName}))
5fc7a1 ▷   {
cfea3a ▷   ▷   return $vars{$varName};
259371 ▷   }
7Ødc Ø2 ▷   elsif (!$undefOkay)
dØc7a1 ▷   {
99cØ5Ø ▷   ▷   &Error("Undefined variable '$varName' in $fileName line $.");
329371 ▷   }
37efe6 }
fdaf5a
f7faaa sub NullFilter
a1bb36 {
dc9bb4 ▷   Ø;
cdefe6 }
9daf5a
3dcafc sub IfFilter
b7bb36 {
1252fd ▷   local $_ = $_[Ø];
76af5a
df89d1 ▷   if (/^##else(\s+.*)?/)
17c7a1 ▷   {
e6e1Ø2 ▷   ▷   return 1;
d39371 ▷   }
11e55c ▷   elsif (/^##endif(\s+.*)?/)
68c7a1 ▷   {
adØe66 ▷   ▷   return 2;
c89371 ▷   }
aØØe24 ▷   else
a2c7a1 ▷   {
41bbde ▷   ▷   return Ø;
6b9371 ▷   }
47efe6 }
d2af5a
bØb322 sub DoFile
4cbb36 {
277744 ····local $fileName = $_[Ø];
39bb62 ▷   my $path;
13Ø8e9 ▷   local *FILE;
ddaf5a
829ea7 ▷   if ($fileName =~ m¦^/¦)
5bc7a1 ▷   {
cc778Ø ▷   ▷   $path = $fileName;
559371 ▷   }
31Øe24 ▷   else
8ac7a1 ▷   {
82b6bc ▷   ▷   for $dir (@incPath)
8a5b34 ▷   ▷   {
e12ae1 ▷   ▷   ▷   if (-e "$dir/$fileName")
f7f776 ▷   ▷   ▷   {
d913cØ ▷   ▷   ▷   ▷   $path = "$dir/$fileName";
9cf22b ▷   ▷   ▷   ▷   last;
8da3a6 ▷   ▷   ▷   }
```

```
b5Øfe4 ▷   ▷   }
6b9371 ▷   }
5f5cc7 ▷   if ($path eq "")
41c7a1 ▷   {
e41cØe ▷   ▷   &Error("Can't find '$fileName', from $fileName line $.");
b59371 ▷   }
e9af5a
6Ø5cba ▷   open(FILE, "<$path") ¦¦ &Error("Can't open $path: $!");
d6Ø169 ▷   &DoOpenFile(*FILE, *NullFilter, Ø);
3bd33d ▷   close(FILE) ¦¦ die;
fc9bb4 ▷   Ø;
caefe6 }
fcaf5a
c9d7db sub DoPrepass
c6bb36 {
d9ac3c ▷   local ($_, $skipFlag) = @_;
d3af5a
Øad534 ▷   return "" if /^###/;
93a9fØ ▷   s/\s*###.*//;▷  ▷   ▷   ▷   ▷   ▷   ▷   ▷   # Strip comments
b34e1b ▷   s/\$\{(\w+)}/&VarSubst($1, $skipFlag)/eg;▷  # Do variable substitutions
372475 ▷   $_;
c5efe6 }
37af5a
f8b89f sub DoOpenFile
9cbb36 {
b17cØc ▷   local *FILE = $_[Ø];
3dØc93 ▷   local *filter = $_[1];
d27f75 ▷   my $skipFlag = $_[2];
c5Ø2e5 ▷   my $result;
3672d5 ▷   local $_;
61af5a
6d8f9e ▷   while (<FILE>)
93c7a1 ▷   {
388a8Ø ▷   ▷   $_ = &DoPrepass($_, $skipFlag);
7cad1d ▷   ▷   if ($result = &filter($_))
Øe5b34 ▷   ▷   {
56cd7c ▷   ▷   ▷   return $result;
26Øfe4 ▷   ▷   }
fdc6f6 ▷   ▷   elsif (/^##(\w*)(\s+(.*))?/)
255b34 ▷   ▷   {
6d7e6b ▷   ▷   ▷   my ($cmd, $params) = ($1, $3);
b5af5a
b98319 ▷   ▷   ▷   if ($cmd =~ /^if/)
8af776 ▷   ▷   ▷   {
8a7f42 ▷   ▷   ▷   ▷   my $condition;
6e7ed1 ▷   ▷   ▷   ▷   my $ifStartLine = $.;
f4af5a
1Ø88Øf ▷   ▷   ▷   ▷   if ($cmd eq "if")
Øa72c3 ▷   ▷   ▷   ▷   {
596eØf ▷   ▷   ▷   ▷   ▷   if ($params =~ /^(\d+)\s*$/)
374859 ▷   ▷   ▷   ▷   ▷   {
2375b5 ▷   ▷   ▷   ▷   ▷   ▷   $condition = int($1);
6b1c89 ▷   ▷   ▷   ▷   ▷   }
91d7f9 ▷   ▷   ▷   ▷   ▷   elsif ($params =~ /^(\d+)\s*([=!]=¦[<>]=?)\s*(\d+)\s*$/)
774859 ▷   ▷   ▷   ▷   ▷   {
6661dc ▷   ▷   ▷   ▷   ▷   ▷   my ($left, $op, $right) = ($1, $2, $3);
41af5a
7461a3 ▷   ▷   ▷   ▷   ▷   ▷   $condition = eval($left . $op . $right);
Ø91c89 ▷   ▷   ▷   ▷   ▷   }
Ø6d76e ▷   ▷   ▷   ▷   ▷   elsif ($params =~ /^(\S+)\s*(eq¦ne)\s*(\S+)\s*$/)
3c4859 ▷   ▷   ▷   ▷   ▷   {
Ø761dc ▷   ▷   ▷   ▷   ▷   ▷   my ($left, $op, $right) = ($1, $2, $3);
5daf5a
Øc96ef ▷   ▷   ▷   ▷   ▷   ▷   $left =~ s/([\\'])/\\$1/g;
bfecd1 ▷   ▷   ▷   ▷   ▷   ▷   $right =~ s/([\\'])/\\$1/g;
Ø5Ø4d4 ▷   ▷   ▷   ▷   ▷   ▷   $condition = eval("'$left' $op '$right'");
bf1c89 ▷   ▷   ▷   ▷   ▷   }
c34bd6 ▷   ▷   ▷   ▷   ▷   else
6e4859 ▷   ▷   ▷   ▷   ▷   {
3cØØ3c ▷   ▷   ▷   ▷   ▷   ▷   &Error("Invalid ##if params: '$params' " .
28e885 ▷   ▷   ▷   ▷   ▷   ▷   ▷   ```"in $fileName line $.");
4d1c89 ▷   ▷   ▷   ▷   ▷   }
```

```
862613 ▷   ▷   ▷   ▷   }
b557e1 ▷   ▷   ▷   ▷   elsif ($cmd =~ /^ifn?def$/)
a772c3 ▷   ▷   ▷   ▷   {
4ac834 ▷   ▷   ▷   ▷   ▷   if ($params =~ /^(\w+)\s*$/)
154859 ▷   ▷   ▷   ▷   ▷   {
f6f1d1 ▷   ▷   ▷   ▷   ▷   ▷   $condition = defined($vars{$1});
ba282a ▷   ▷   ▷   ▷   ▷   ▷   $condition = !$condition if ($cmd eq "ifndef");
b21c89 ▷   ▷   ▷   ▷   ▷   }
684bd6 ▷   ▷   ▷   ▷   ▷   else
514859 ▷   ▷   ▷   ▷   ▷   {
feaa2Ø ▷   ▷   ▷   ▷   ▷   ▷   &Error("Invalid ##$cmd param: '$params' " .
48e885 ▷   ▷   ▷   ▷   ▷   ▷   ▷   "in $fileName line $.");
d21c89 ▷   ▷   ▷   ▷   ▷   }
172613 ▷   ▷   ▷   ▷   }
63af5a
ab38fd ▷   ▷   ▷   ▷   # Do main body of if
88e33Ø ▷   ▷   ▷   ▷   $result = &DoOpenFile(*FILE, *IfFilter,
ØØØ7dd ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   $skipFlag || !$condition);
5aaf5a
Ø322eØ ▷   ▷   ▷   ▷   if ($result == 1)▷  # an '##else' was found
9472c3 ▷   ▷   ▷   ▷   {
494be6 ▷   ▷   ▷   ▷   ▷   # Handle else
8e6c67 ▷   ▷   ▷   ▷   ▷   $result = &DoOpenFile(*FILE, *IfFilter,
834955 ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   $skipFlag || $condition);
732613 ▷   ▷   ▷   ▷   }
8eaf5a
f8c39f ▷   ▷   ▷   ▷   if ($result == 1)▷  # a second '##else' was found
8772c3 ▷   ▷   ▷   ▷   {
fØa3ab ▷   ▷   ▷   ▷   ▷   &Error("Two ##else's in a row in $fileName line $.");
592613 ▷   ▷   ▷   ▷   }
4Ø6Ø47 ▷   ▷   ▷   ▷   elsif ($result == Ø)▷   # EOF was encountered
Øb72c3 ▷   ▷   ▷   ▷   {
5ff612 ▷   ▷   ▷   ▷   ▷   &Error("Unterminated ##if " .
e Ø1b53 ▷   ▷   ▷   ▷   ▷   ▷   "in $fileName line $ifStartLine");
8d2613 ▷   ▷   ▷   ▷   }
Øfa3a6 ▷   ▷   ▷   }
e3c4c2 ▷   ▷   ▷   elsif ($cmd eq "include")
aØf776 ▷   ▷   ▷   {
Ø536cc ▷   ▷   ▷   ▷   if ($skipFlag)
6672c3 ▷   ▷   ▷   ▷   {
542613 ▷   ▷   ▷   ▷   }
9e56a3 ▷   ▷   ▷   ▷   elsif ($params =~ /^"(.*)"\s*$/)
Ø572c3 ▷   ▷   ▷   ▷   {
2c2181 ▷   ▷   ▷   ▷   ▷   my $incFile = $1;
85af5a
f5e259 ▷   ▷   ▷   ▷   ▷   &DoFile($incFile);
bf2613 ▷   ▷   ▷   ▷   }
cefaØ7 ▷   ▷   ▷   ▷   else
fa72c3 ▷   ▷   ▷   ▷   {
bØ9Ø76 ▷   ▷   ▷   ▷   ▷   &Error("Invalid ##include params: '$params'");
172613 ▷   ▷   ▷   ▷   }
3Øa3a6 ▷   ▷   ▷   }
1395b4 ▷   ▷   ▷   elsif ($cmd eq "set")
f6f776 ▷   ▷   ▷   {
22e188 ▷   ▷   ▷   ▷   if ($params =~ /^(\w+)=<<(")(.*)"\s*$/ or
1e74dc ▷   ▷   ▷   ▷   ▷   $params =~ /^(\w+)=<<(')(.*)'\s*$/)
9d72c3 ▷   ▷   ▷   ▷   {
2f556a ▷   ▷   ▷   ▷   ▷   my $varName = $1;
87e87b ▷   ▷   ▷   ▷   ▷   my $quoteChar = $2;
53b48c ▷   ▷   ▷   ▷   ▷   my $endTag = $3 . "\n";
caØ277 ▷   ▷   ▷   ▷   ▷   my $value;
e1af5a
2acec1 ▷   ▷   ▷   ▷   ▷   while (<FILE>)
ae4859 ▷   ▷   ▷   ▷   ▷   {
1dbf65 ▷   ▷   ▷   ▷   ▷   ▷   if ($_ eq $endTag)
328fec ▷   ▷   ▷   ▷   ▷   ▷   {
e1fe1d ▷   ▷   ▷   ▷   ▷   ▷   ▷   chop $value;
8Ø8bc2 ▷   ▷   ▷   ▷   ▷   ▷   ▷   last;
Øedb3c ▷   ▷   ▷   ▷   ▷   ▷   }
bØ7f8b ▷   ▷   ▷   ▷   ▷   ▷   else
2b8fec ▷   ▷   ▷   ▷   ▷   ▷   {
8775Ø4 ▷   ▷   ▷   ▷   ▷   ▷   ▷   if ($quoteChar eq '"')
```

```
26Ø6b8 ▷   ▷   ▷   ▷   ▷   ▷   ▷   {
3e921Ø ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   $␣ = &DoPrepass($␣, $skipFlag);
115268 ▷   ▷   ▷   ▷   ▷   ▷   ▷   }
363eØ5 ▷   ▷   ▷   ▷   ▷   ▷   ▷   $value .= $␣;
22db3c ▷   ▷   ▷   ▷   ▷   ▷   }
ae1c89 ▷   ▷   ▷   ▷   ▷   }
9f69be ▷   ▷   ▷   ▷   ▷   if (!$skipFlag)
Ø84859 ▷   ▷   ▷   ▷   ▷   {
2758a8 ▷   ▷   ▷   ▷   ▷   ▷   $vars{$varName} = $value;
381c89 ▷   ▷   ▷   ▷   ▷   }
d82613 ▷   ▷   ▷   ▷   }
ba2aØ7 ▷   ▷   ▷   ▷   elsif ($params =~ /^(\w+)="(.*)"\s*$/ or
56484b ▷   ▷   ▷   ▷   ▷   ▷▷▷$params =~ /^(\w+)=(\S*)\s*$/)
ec72c3 ▷   ▷   ▷   ▷   {
e969be ▷   ▷   ▷   ▷   ▷   if (!$skipFlag)
a24859 ▷   ▷   ▷   ▷   ▷   {
Øc1359 ▷   ▷   ▷   ▷   ▷   ▷   $vars{$1} = $2;
731c89 ▷   ▷   ▷   ▷   ▷   }
842613 ▷   ▷   ▷   ▷   }
9efaØ7 ▷   ▷   ▷   ▷   else
bc72c3 ▷   ▷   ▷   ▷   {
947761 ▷   ▷   ▷   ▷   ▷   &Error("Invalid ##set command: '$params'");
f62613 ▷   ▷   ▷   ▷   }
ffa3a6 ▷   ▷   ▷   }
84bb8c ▷   ▷   ▷   else
54f776 ▷   ▷   ▷   {
3fa974 ▷   ▷   ▷   ▷   &Error("Unrecognized command: '$␣'");
a6a3a6 ▷   ▷   ▷   }
51Øfe4 ▷   ▷   }
816ac8 ▷   ▷   elsif (!$skipFlag)
3f5b34 ▷   ▷   {
214522 ▷   ▷   ▷   print;
2ØØfe4 ▷   ▷   }
b99371 ▷   }
2d3e6b ▷   return Ø;
56efe6 }
a5af5a
616b86 $optEnable = 1;
e9af5a
27a355 foreach (@ARGV)
dabb36 {
7a2f2b ▷   if ($optEnable and /^-/)
dfc7a1 ▷   {
d93Ød Ø ▷   ▷   if (/^--$/)
9a5b34 ▷   ▷   {
6Ø2ab8 ▷   ▷   ▷   $optEnable = Ø;
acØfe4 ▷   ▷   }
1244ee ▷   ▷   elsif (/^-D(\w+)=(.*)$/)
f55b34 ▷   ▷   {
e2fØae ▷   ▷   ▷   $vars{$1} = $2;
eaØfe4 ▷   ▷   }
1bb97d ▷   ▷   elsif (/^-I(.*)$/)
1Ø5b34 ▷   ▷   {
Øf1ff6 ▷   ▷   ▷   unshift @incPath, $1;
28Øfe4 ▷   ▷   }
1dØe1d ▷   ▷   else
355b34 ▷   ▷   {
Ø9217a ▷   ▷   ▷   &Error("Unrecognized option: '$␣'");
d3Øfe4 ▷   ▷   }
239371 ▷   }
a3Øe24 ▷   else
e5c7a1 ▷   {
42f723 ▷   ▷   &DoFile($␣);
5f9371 ▷   }
3eefe6 }
8faf5a
2da6Ø1 #
92f1a9 # vi: ai ts=4
be887e # vim: si
5ea6Ø1 #
```

```
0b326b #!/usr/bin/env perl
44a601 #
40eb94 # psgen -- Postscript generator for code portion of source books
e6a601 #
63b470 # Reads in a list of files/dirs from <filelist>, runs munge on each of
5a082c # them, and generates a single postscript file to stdout.  The page numbers
bb7bb6 # for each file/dir are put into the file <pagenums>.
c3a601 #
d95c67 # usage: psgen [ options... ] <filelist> <pagenums> <volume #> > foo.ps
c96689 #>  >   >   -l<firstLogicalPage>
e0d7bf #>  >   >   -p<firstPhysicalPage>
f8e733 #>  >   >   -f<font>
01e27b #>  >   >   -D<defs> (passed to yapp)
a28b57 #>  >   >   -P<productNumber>
52dd5b #>  >   >   -o<mungedOutFile>
9d1999 #>  >   >   -e> >   >   >   (auto edit errors)
47a601 #
3014bc # $Id: psgen,v 1.18 1997/11/13 21:44:16 colin Exp $
53a601 #
68af5a
a3aa99 use File::Basename;
10af5a
97f0d2 $bookRoot = $ENV{"BOOKROOT"} || ".";
37858c $toolsDir = dirname(__FILE__);
766cd4 $psDir = "$bookRoot/ps";
8da30a $editor = $ENV{"EDITOR"} || "vi";
a5af5a
67a784 # Configuration settings - external file names
978791 $mungeProg = "$toolsDir/munge";
5b416e $yappProg = "$toolsDir/yapp";
3c9bf2 $preambleFile = "$psDir/prolog.ps";
ecac62 $tempFile = "/tmp/psgen-$$";
bbaf5a
a5931f # Parse arguments
c53168 $firstLogPage = $firstPhysPage = 0;
02b2fb $productNumber = 1;
23d31d $font = "Inconsolata";
2f06f0 $autoEdit = 0;
216bf8 while ($#ARGV >= 0 && $ARGV[0] =~ /^-/)
54bb36 {
e0c41b >   $_ = shift @ARGV;
1fa7a3 >   if (/^--$/)
f0c7a1 >   {
244d04 >   >   last;
b79371 >   }
a6686e >   elsif (/^-l(\d+)$/)
c5c7a1 >   {
fd9768 >   >   $firstLogPage = $1;
829371 >   }
c96389 >   elsif (/^-p(\d+)$/)
f9c7a1 >   {
a9daca >   >   $firstPhysPage = $1;
2f9371 >   }
943ea6 >   elsif (/^-f(.+)$/)
b4c7a1 >   {
d45047 >   >   $font = $1;
449371 >   }
545265 >   elsif (/^-D(.+)$/)
9ec7a1 >   {
d2b5b5 >   >   $yappDefs .= " " . $_;
d09371 >   }
38cf3c >   elsif (/^-P(\d+)$/)
c5c7a1 >   {
4612ce >   >   $productNumber = $1;
ec9371 >   }
740c91 >   elsif (/^-o(.+)$/)
f5c7a1 >   {
b598f0 >   >   $mungedOutFile = $1;
579371 >   }
1e0a96 >   elsif (/^-e$/)
b1c7a1 >   {
6a8566 >   >   $autoEdit = 1;
```

```
a19371 ▷   }
ØeØe24 ▷   else
96c7a1 ▷   {
4d73e7 ▷   ▷   &Error("Unrecognized option: '$␣'");
d89371 ▷   }
7eefe6 }
ea622b $fileListFile = shift @ARGV ¦¦ die "Missing file list argument (arg 1)";
7Ø37cc $pageNumFile = shift @ARGV ¦¦ die "Missing page number file argument (arg 2)";
ae1424 $volume = shift @ARGV ¦¦ die "Missing volume number argument (arg 3)";
ffaf5a
1bb12a # Determine initial page numbers
33bb36 {
2696Ø8 ▷   my $nextLogPage = 1;
Øf36e6 ▷   my $nextPhysPage = 3;
24fb6f ▷   my $volNum = Ø;▷▷   # Which volume's page numbers we're reading
51af5a
bc46f3 ▷   if ($volume > 1)
7fc7a1 ▷   {
487997 ▷   ▷   open(OLDPAGENUMS, "<$pageNumFile") ¦¦ die;
62ff73 ▷   ▷   while (<OLDPAGENUMS>)
615b34 ▷   ▷   {
b7d954 ▷   ▷   ▷   if (/^Volume\s+(\d+)$/)
e1f776 ▷   ▷   ▷   {
e3Ø7d9 ▷   ▷   ▷   ▷   $volNum = $1;
65a3a6 ▷   ▷   ▷   }
d586Ø7 ▷   ▷   ▷   elsif (/^Next:\s+(\d+)\s*$/ && $volNum == $volume - 1)
c2f776 ▷   ▷   ▷   {
eØ92ae ▷   ▷   ▷   ▷   $nextLogPage = $1;
caa3a6 ▷   ▷   ▷   }
7dØfe4 ▷   ▷   }
e9c87a ▷   ▷   close(OLDPAGENUMS);
e49371 ▷   }
adØe24 ▷   else
66c7a1 ▷   {
e2e1c5 ▷   ▷   unlink($pageNumFile);
7b9371 ▷   }
d7c737 ▷   $firstLogPage = $nextLogPage if ($firstLogPage == Ø);
7c9133 ▷   $firstPhysPage = $nextPhysPage if ($firstPhysPage == Ø);
65efe6 }
58af5a
6cfcdØ # Names of PostScript operators invoked. ‹These are the interface
646fa9 # between this file and the $preambleFile.
1ee2Øe $oddPageStartPS = "OddPageStart";
6a3472 $evenPageStartPS = "EvenPageStart";
19d55c $oddPageEndPS = "OddPageEnd";
Ø7e5cf $evenPageEndPS = "EvenPageEnd";
a71aa5 $dirPagePS = "DirPage";
414ce4 # This is short because it's emitted every line
13Øfe1 $linePS = "L";
37af5a
84c132 # Handle an error from munge.
27d359 # A result of Ø means to retry, 1 means to exit
e5aaee sub MungeError
5fbb36 {
8e4a8c ▷   my $result = 1;
52af5a
2bØ5b8 ▷   open(FILEH, "<$tempFile") ¦¦ die;
27c336 ▷   while (<FILEH>)
7cc7a1 ▷   {
e18ed1 ▷   ▷   print STDERR;
d993Ø3 ▷   ▷   if (/ in (.*) line (\d+)$/)
9a5b34 ▷   ▷   {
13aef5 ▷   ▷   ▷   my ($fileName, $lineNumber) = ($1, $2);
e8af5a
311855 ▷   ▷   ▷   if ($autoEdit)
b3f776 ▷   ▷   ▷   {
1382bb ▷   ▷   ▷   ▷   my @statResult = stat($fileName);
1b41c4 ▷   ▷   ▷   ▷   my $oldMTime = $statResult[9];
7faf5a
c8aaØa ▷   ▷   ▷   ▷   system("'$editor' '+$lineNumber' '$fileName' 1>&2");
4b7452 ▷   ▷   ▷   ▷   @statResult = stat($fileName);
bed189 ▷   ▷   ▷   ▷   $result = ($statResult[9] == $oldMTime);
```

```
3af22b ▷   ▷   ▷   ▷   last;
b2a3a6 ▷   ▷   ▷   }
7aØfe4 ▷   ▷   }
Øf9371 ▷   }
aeØb1b ▷   close(FILEH);
31867c ▷   unlink($tempFile) ¦¦ die "Couldn't unlink $tempFile";
5741ed ▷   return $result;
8befe6 }
4daf5a
47838b sub CopyFileToPS
a6bb36 {
b5d166 ▷   local $fileName = $_[Ø];
ee7d18 ▷   local $args = "'-I$psDir' '-Dfont=$font'";
dc72d5 ▷   local $_;
5faf5a
1ØØ6Ø5 ▷   $args .= $yappDefs;
7bd5ff ▷   open(FILEH, "$yappProg $args '$fileName' ¦") ¦¦ die;
cec336 ▷   while (<FILEH>)
95c7a1 ▷   {
ad896c ▷   ▷   print PSOUT $_;
939371 ▷   }
fcd7Ød ▷   close(FILEH) ¦¦ exit(1);
5ac168 ▷   1;
f6efe6 }
49af5a
e78ea9 # Wrap a string in parens as required by PostScript, with proper quoting.
944d6f sub StringPS
5abb36 {
1ceacf ▷   local $str = $_[Ø];
d6af5a
9af383 ▷   $str =~ s/([\\()])/\\$1/g;
75ad4b ▷   "(" . $str . ")";
7Øefe6 }
9faf5a
8Øf47Ø # Emit a start of page. `The Postscript DSC %%Page: header`
9e235a # (followed by logical page number, then physical) and
12d1c3 # the top-of-page function (which is passed the page number as a string)
9789cØ sub PageStartPS
42bb36 {
cdb425 ▷   local $pageNum = $_[Ø];
1faf5a
911756 ▷   "%%Page: " . ($pageNum + $firstLogPage) . " " .
5520f1 ▷   ▷   ▷   ▷   `($pageNum + $firstPhysPage) . "\n" .
34eØ1e ▷   ▷   &StringPS($pageNum + $firstLogPage) .
52120d ▷   ▷   (($pageNum + $firstLogPage) % 2) ? $oddPageStartPS
Ø9Ø127 ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ▷   ``: $evenPageStartPS) . "\n";
81efe6 }
92af5a
7d9c34 sub PageEndPS
Øbbb36 {
92b425 ▷   local $pageNum = $_[Ø];
Ø1af5a
1dceee ▷   (($pageNum + $firstLogPage) % 2) ? $oddPageEndPS : $evenPageEndPS) . "\n";
d5efe6 }
a9af5a
adadcØ # Save the page number to a table-of-contents file
3225ef sub SavePageNum
a1bb36 {
b976a9 ▷   local ($fileName, $pageNum) = @_;
bbaf5a
ba17c4 ▷   print PAGENUMS ($pageNum + $firstLogPage), ": $fileName\n";
ccefe6 }
3aaf5a
39f73e # The main code.
2caf5a
82431Ø open(PSOUT, ">-") ¦¦ die;
efae9a open(FILELIST, "<$fileListFile") ¦¦ die;
ce8Ødf open(PAGENUMS, ">>$pageNumFile") ¦¦ die;
6d5c67 if ($mungedOutFile ne "")
45bb36 {
549d6f ▷   open(MUNGEDOUT, ">$mungedOutFile") ¦¦ die;
41efe6 }
```

```
e0af5a
813222 print PAGENUMS "Volume $volume\n";
a0af5a
18139Ø &CopyFileToPS($preambleFile);
e4af5a
e87633 $fileNumber = Ø;
7Ø34fØ $pageNum = Ø;▷  # This is Ø-based, since it is added to $first{Log,Phys}Page
bbØ7d1 $enable = Ø;
cfaf5a
c8d231 while (<FILELIST>)
99bb36 {
28ec5e ▷   /^([VDTB])(\S*)\s+(.*)/ ¦¦ die "Illegal file list line $.";
87af5a
f19a4e ▷   local ($fileType, $options, $arg) = ($1, $2, $3);
7daf5a
29ef81 ▷   if ($fileType eq "V")
bfc7a1 ▷   {
9a7ac4 ▷   ▷   @args = split(/\s+/, $arg);
9d43d2 ▷   ▷   if ($enable = ($args[Ø] == $volume))
865b34 ▷   ▷   {
c46632 ▷   ▷   ▷   $defaultTabWidth = int($args[1]);
82Øfe4 ▷   ▷   }
a49371 ▷   }
6fØ3f6 ▷   elsif ($fileType eq "D")
62c7a1 ▷   {
cØ9557 ▷   ▷   next unless $enable;▷   # Do nothing if we're in the wrong volume
dfee1c ▷   ▷   $dirName = $arg;
5343f5 ▷   ▷   &SavePageNum($dirName, $pageNum);
1a54d1 ▷   ▷   print PSOUT &PageStartPS($pageNum);
ce1122 ▷   ▷   print PSOUT &StringPS($dirName), $dirPagePS, "\n";
9Ø5dØ9 ▷   ▷   print PSOUT &PageEndPS($pageNum);
fd4513 ▷   ▷   $pageNum++;
aa9371 ▷   }
ee0e24 ▷   else
4ac7a1 ▷   {
85ac26 ▷   ▷   my $done = Ø;
74af5a
8843dØ ▷   ▷   $fileNumber++;
5ace26 ▷   ▷   $fileName = $arg;
139557 ▷   ▷   next unless $enable;▷   # Do nothing if we're in the wrong volume
46b29c ▷   ▷   &SavePageNum($fileName, $pageNum);
329435 ▷   ▷   $quotedFileName = $fileName;
bdb4e1 ▷   ▷   $quotedFileName =~ s/'/\\'/g;
3b9Ø9Ø ▷   ▷   $tabWidth = ($options =~ /(\d)/) ? $1 : $defaultTabWidth;
bdde2e ▷   ▷   $args = ($fileType eq "B") ? "-b" : "";
4811ea ▷   ▷   $args .= " -$tabWidth -p$productNumber -f$fileNumber";
a5ee0d ▷   ▷   while (!$done)
795b34 ▷   ▷   {
8dØ9eb ▷   ▷   ▷   if (open(FILE, "$mungeProg $args '$quotedFileName' 2>$tempFile ¦"))
f4f776 ▷   ▷   ▷   {
9142fØ ▷   ▷   ▷   ▷   $line = <FILE>;
b6ae88 ▷   ▷   ▷   ▷   print MUNGEDOUT $line;
daaf5a
Ø54fff ▷   ▷   ▷   ▷   while ($line ne "")
fc72c3 ▷   ▷   ▷   ▷   {
9e8f6c ▷   ▷   ▷   ▷   ▷   print PSOUT &PageStartPS($pageNum);
Ø1af5a
65e9df ▷   ▷   ▷   ▷   ▷   while ($line ne "" and $line !~ /^\f/)
6a4859 ▷   ▷   ▷   ▷   ▷   {
85fba6 ▷   ▷   ▷   ▷   ▷   ▷   chop $line;
Øa8b78 ▷   ▷   ▷   ▷   ▷   ▷   print PSOUT &StringPS($line), $linePS, "\n";
22Øc59 ▷   ▷   ▷   ▷   ▷   ▷   $line = <FILE>;
17315c ▷   ▷   ▷   ▷   ▷   ▷   print MUNGEDOUT $line;
4a1c89 ▷   ▷   ▷   ▷   ▷   }
ØØØ47a ▷   ▷   ▷   ▷   ▷   $line =~ s/^\f//;
81af5a
6a85f3 ▷   ▷   ▷   ▷   ▷   print PSOUT &PageEndPS($pageNum);
5164dc ▷   ▷   ▷   ▷   ▷   $pageNum++;
eØ2613 ▷   ▷   ▷   ▷   }
49af5a
3eb28e ▷   ▷   ▷   ▷   if (close(FILE))
7572c3 ▷   ▷   ▷   ▷   {
```

```
c2e33c ▷   ▷   ▷   ▷   ▷   $done = 2;
c52613 ▷   ▷   ▷   ▷   }
41faØ7 ▷   ▷   ▷   ▷   else
9372c3 ▷   ▷   ▷   ▷   {
956Ø22 ▷   ▷   ▷   ▷   ▷   $done = &MungeError();
5c2613 ▷   ▷   ▷   ▷   }
Øea3a6 ▷   ▷   ▷   }
bfbb8c ▷   ▷   ▷   else
7ff776 ▷   ▷   ▷   {
576e6e ▷   ▷   ▷   ▷   $done = &MungeError();
2ba3a6 ▷   ▷   ▷   }
74Øfe4 ▷   ▷   }
15Ød39 ▷   ▷   if ($done == 1)
5c5b34 ▷   ▷   {
65c434 ▷   ▷   ▷   die;
3fØfe4 ▷   ▷   }
6c9371 ▷   }
Ø9efe6 }
25af5a
2e5aØc # Print PostScript DSC trailer with the correct number of pages
4ffdf2 print PSOUT "%%Trailer\n%%Pages: ", $pageNum, "\n%%EOF\n";
98af5a
e3f624 print PAGENUMS "Pages: ", $pageNum, "\n";
26Ø7e5 print PAGENUMS "Next: ", ((($pageNum+1) & ~1) + $firstLogPage), "\n";
35af5a
eeØ95d close(PAGENUMS) ¦¦ die;
8f93f8 close(FILELIST) ¦¦ die;
d7c2eØ close(PSOUT) ¦¦ die;
9Øaf5a
5b5c67 if ($mungedOutFile ne "")
bfbb36 {
764d7c ▷   close(MUNGEDOUT) ¦¦ die;
c4efe6 }
7caf5a
b5a6Ø1 #
c9f1a9 # vi: ai ts=4
dd887e # vim: si
Ø1a6Ø1 #
```

```
Øb326b #!/usr/bin/env perl
9eaf5a
71b379 $fileNum = Ø;
Ø95Ø61 while(<>)
debb36 {
5Ø65dc ▷   /^([VDTB])(\S*)\s+(.*)/ ¦¦ die("Bad filelist, line $.");
a3Øc47 ▷   ($type, $options, $name) = ($1, $2, $3);
f6af5a
f34d71 ▷   if ($type eq "D")
cec7a1 ▷   {
66fbØd ▷   ▷   $dir = $name;
7231Ø5 ▷   ▷   print "D $dir\n";
729371 ▷   }
1e83ee ▷   elsif ($type eq "V")
74c7a1 ▷   {
add213 ▷   ▷   # Do nothing
3e9371 ▷   }
5bØe24 ▷   else
b7c7a1 ▷   {
86Ø1c5 ▷   ▷   $fileNum++;
294631 ▷   ▷   $tail = $name;
138774 ▷   ▷   $tail =~ s¦^.*/¦¦;
fØac83 ▷   ▷   die("Bad filelist, line $.") if $name ne $dir . $tail;
a92f88 ▷   ▷   print "$fileNum $tail\n";
1d9371 ▷   }
24efe6 }
ceaf5a
2aa6Ø1 #
9ef1a9 # vi: ai ts=4
5e887e # vim: si
1ba6Ø1 #
```

Directory

books/ps/

```
--9f55 ØØØ8c97d1ef8ØØ2ØØ15 Page 1 of prolog.ps

6666ea ##set pageNumFont="Beteckna"
84e7a5 ##set dirNameFont="Beteckna-Bold"
458b91 ##set fontsNeeded="${font} Beteckna Beteckna-Bold"
a1619b ##set includeFontComments=<<"END"
94e1e3 %%IncludeResource: font ${font}
55e942 %%IncludeResource: font Beteckna
def2c4 %%IncludeResource: font Beteckna-Bold
7aeØd1 END
3db7e1 ##if ${font} eq Courier
9fd7Øf ##set charShrinkFactor=Ø.93
cbdb2d ##set zeroGlyph=Oslash
9578b4 ##set underscoreGlyph=underscore
7c5c8b ##set bulletGlyph=bullet
eef323 ##set tabGlyph=currency
ØØ616a ##set leftQuoteGlyph=quoteleft
7b58b1 ##set rightQuoteGlyph=quoteright
53Ø77Ø ##set pilcrowGlyph=paragraph
f694cb ##set barGlyph=bar
291Øcc ##else
279811 ##set charShrinkFactor=1
4bdb2d ##set zeroGlyph=Oslash
7c24ef ##set underscoreGlyph=underscore2
c2edbf ##set bulletGlyph=bullet2
e68ade ##set tabGlyph=tabsym
1c9d6a ##set leftQuoteGlyph=grave
ddd94Ø ##set rightQuoteGlyph=quoteright▷       ### was "acute"
3aØ142 ##set pilcrowGlyph=erase
76698c ##set barGlyph=orsym
6b1Ø88 ##set do_custom_chars=1
684874 ##endif
5a1176 %!PS-Adobe-3.Ø
1eØ485 %%Orientation: Portrait
5fdfc2 %%Pages: (atend)
3Øa66c %%DocumentNeededResources: font ${fontsNeeded}
e3ea16 %%DocumentMedia: Letter 612 792 74 white ()
af3e6b %%EndComments
15b6af %%BeginDefaults
18Ø565 %%PageMedia: Letter
1535ce %%PageResources: font ${fontsNeeded}
Ø93b1a %%EndDefaults
ffb415 %%BeginProlog
b8238f %%BeginResource: procset Custom-Preamble Ø Ø
25f2d1 %
7ecdda % Document definitions
b1162d % (Upper case to avoid collisions)
9ef2d1 %
c3af5a
d1833d % 8.5x11 paper is 612x792 points, but 24 points near the edge or so
88afbØ % shouldn't be used.
fe16ec /Topmargin 77Ø def
ff5cb5 /Leftmargin 3Ø def
216444 /Rightmargin 612 Leftmargin sub def
adfØe7 /Botmargin 22 def
994cce /Bindoffset 4Ø def
38af5a
969276 /Lineskip -1Ø def
f2d741 % How much to shrink characters by?
727fb4 /Factor ${charShrinkFactor} def
fb2721 /Fontsize 9.5 Factor mul def
c9Øcc9 % (1ØØØ units is std height, so Courier at 6/1Ø aspect ratio is 6ØØ.
52a64b % Widen to make up for scaling loss.
bae474 /Charwidth
771529 ‘‘Rightmargin Leftmargin sub Bindoffset sub 87 div Fontsize div 1ØØØ mul
d14927 def
8caf5a
7629b3 % Print a header (expects page number on stack)
335cc4 /OddPageStart
eØa239 { save exch /MyFont findfont Fontsize scalefont setfont‘
211584 ‘‘/CurrentLeft Leftmargin Bindoffset add def
2Øda17 ‘‘/CurrentRight Rightmargin def
44ab9Ø ‘‘CurrentLeft Topmargin moveto } def
8caf5a
```

```
--b363 ØØØ41688cd8Ø2ØØ15 Page 2 of prolog.ps

b911a7 /EvenPageStart
29a239 { save exch /MyFont findfont Fontsize scalefont setfont▪
76ba7e ▪▪/CurrentLeft Leftmargin def
fe5e96 ▪▪/CurrentRight Rightmargin Bindoffset sub def
7bab9Ø ▪▪CurrentLeft Topmargin moveto } def
29af5a
52fcac % /MyFont findfont [Fontsize Ø Ø Fontsize Ø Ø] makefont setfont
92af5a
a27196 % Print the name of the directory in a large font
4cf717 /DirPage
63bb36 {
Ø348fa ▪▪/${dirNameFont} findfont 14 scalefont setfont
a3d87a ▪▪Ø -1Ø rmoveto (Directory) show
e519de ▪▪CurrentLeft 3Ø add currentpoint exch pop 2Ø sub moveto show
35d968 } def
7aaf5a
3237d9 % Advance a line
c5bbd9 /L {show CurrentLeft currentpoint exch pop Lineskip add moveto} bind def▪
f2af5a
942faa % Print the "inside" footer line using P (string font => )
a99ab1 % We do some magic involving redefining P to first measure the
cØ7b4c % width of this string and then print it, so you must use it
e73f98 % to do all printing.
b1795a /Foot {
11135a ##ifdef footerFile
dd4bbØ ##include "${footerFile}"
854874 ##endif
abd968 } def
a5af5a
7Ø21f4 % /P is defined in the Setup section
1aaf5a
ed4721 % Print an odd footer
Ø3c8e7 /OddPageEnd
863Ø5b ▪{ CurrentLeft Botmargin moveto CurrentRight Botmargin lineto
275bee ▪▪▪1 setlinewidth stroke
1e9553 ▪▪▪CurrentLeft Botmargin 1Ø sub moveto
8f555b ▪▪▪Foot
422c7a ▪▪▪1Ø string cvs dup stringwidth
7c4778 ▪▪▪pop CurrentRight exch sub currentpoint exch pop moveto
Øfb4b5 ▪▪▪/${pageNumFont} P
e2a46Ø ▪▪▪showpage
a51Ø2b ▪▪▪restore
c8d968 } def
61af5a
b17e6c % Print an even footer
388eØ8 /EvenPageEnd
3b3Ø5b ▪{ CurrentLeft Botmargin moveto CurrentRight Botmargin lineto
c35bee ▪▪▪1 setlinewidth stroke
fe1363 ▪▪▪Leftmargin Botmargin 1Ø sub moveto
2Øc7de ▪▪▪/${pageNumFont} P▪
a23347 ▪▪▪CurrentRight FootWidth sub currentpoint exch pop moveto
1b555b ▪▪▪Foot
eaa46Ø ▪▪▪showpage
241Ø2b ▪▪▪restore
96d968 } def
Ø5af5a
7d1aa9 ##ifdef do_custom_chars
ed5c68 % A 1ØØØ-point OCRB discunderline consists of:
714a61 % 111.45 ▪-173.688 moveto
f3872c % 6Ø9.356 -173.688 lineto
83d4Ø8 % 6Ø9.356 ▪-7Ø.9227 lineto
b3f7be % 111.45 ▪▪-7Ø.9227 lineto
7aaeb6 % closepath
f71baØ % 72Ø.Ø ▪▪▪-Ø.Ø moveto
Øbb9fb % Line thickness is
ebØ64e % 1Ø2.7653 pts.
baaf5a
63a656 % This would suggest the following values:
9a3daa /underleft 111.45 def
a9daed /underright 6Ø9.356 def
7974e6 /underthick 1Ø2.7643 def
Ø4bc2f /underup underthick def
```

```
1ef7a5 /underdown Ø def
e8Ø7eb /underserif 25 def
74af5a
19a1cd % These look better in GhostScript, but not on a real Adobe rasterizer
a4b92f %/underright 6ØØ def
cd5211 %/underleft 1ØØ def
27e37b %/underthick 75 def
62af5a
d45ac2 171
dca9d6 211
4a285a 36Ø81
a5e4c4 % The default bullet character is
57468a % 254.Ø 341.Ø moveto
d2f6eØ % 254.Ø 17Ø.Ø lineto
b99452 % 465.Ø 17Ø.Ø lineto
ff4d77 % 465.Ø 341.Ø lineto
c1aeb6 % closepath
Ø1e4fe % Our modified version is based on:
7c72b6 /bullwid 2Ø4 def
a619ba /bullht 176.75 def
bcfØ55 /bullleft 254 341 add bullwid sub 2 div def
e43492 /bullright 254 341 add bullwid add 2 div def
7e3Øcb /bullbot 254 def
bf5377 /bulltop bullbot bullht add def
94af5a
e1f6bd % And a custom-created tab symbol
c6bd4c /tableft 25Ø def
edØbb5 /tabright 55Ø def
e1855f /tabtop 55Ø def
fØf9Øe /tabbot 5Ø def
ac7a9f /tablinewidth 35 def
f2af5a
ef6e1f % Let's try a vertical bar
Øfcf63 % OCRB defines (¦)
6f9c11 % 411.Ø62 -173.688 moveto
Øb1a66 % 411.Ø62 741.Ø43 lineto
c5e9Ø2 % 3Ø8.297 741.Ø43 lineto
fcdØ8f % 3Ø8.297 -173.688 lineto
f4aeb6 % closepath
77cb9e % 72Ø.Ø -Ø.Ø moveto
198429 /orleft 3Ø8.297 def
5439aØ /orright 411.Ø62 def
ae41d9 /orbot -173.688 def
9Ø85ca /ortop 741.Ø43 def
cb3276 /orbreak 15Ø def▷       % Width of break
e86a64 /orbbot ortop orbot add orbreak sub 2 div def▷  % Bottom of break
4de5a1 /orbtop ortop orbot add orbreak add 2 div def▷  % Top of break
8c4874 ##endif
7aaf5a
1e59Øa % newfontname encoding-vec fontname -> -▷       make a new encoded font
bb7a4a /MF2 {
b1a1fe ‘‘% Make a dict for the new font, with room for the /Metrics
d613Ø5 ‘‘findfont dup length 1 add dict begin
d15Ødf ‘‘% Copy everything except the FID entry
df8213 ‘‘{1 index /FID eq {pop pop} {def} ifelse} forall
dc45fa ‘‘% Set the encoding vector
bb4a76 ‘‘/Encoding exch def
17af5a
Øc1aa9 ##ifdef do_custom_chars
dacea1 ‘‘% Create a new expanded CharStrings dictionary
839841 ‘‘CharStrings dup length 5 add dict
48f159 ‘‘begin { def } forall
dd7279 ‘‘% Create a custom underscore character
1e94f9 ‘‘/underscore2 {
7c43d3 ▷       pop
96aa51 ▷       //Charwidth Ø % width, bounding box follows
98b8ac ▷       //underleft //underdown neg //underright //underthick //underup add
5e4b1Ø ▷       setcachedevice
ØfØ49d ▷       //underleft //underthick //underup add moveto
4d43bc ▷       //underleft //underserif add //underthick //underup add lineto
97ae7c ▷       //underleft //underserif add //underthick lineto
4424be ▷       //underright //underserif sub //underthick lineto
```

```
1e9b49 ▷       //underright //underserif sub //underthick //underup add lineto
77d29e ▷       //underright //underthick //underup add lineto
b35e94 ▷       //underright //underdown neg lineto
13afØ1 ▷       //underright //underserif sub //underdown neg lineto
9Ø261d ▷       //underright //underserif sub Ø lineto
9c9724 ▷       //underleft //underserif add Ø lineto
7f619f ▷       //underleft //underserif add //underdown neg lineto
eaf1ef ▷       //underleft //underdown neg lineto
f38a27 ▷       closepath fill
f2Øf1d ▪▪} bind def
f8477a ▪▪% Create a custom bullet character.
6a7f11 ▪▪/bullet2 {
8943d3 ▷       pop
Ø7aa51 ▷       //Charwidth Ø % width, bounding box follows
eadb5d ▷       //bullleft //bullbot //bullright //bulltop
944b1Ø ▷       setcachedevice
a74dfe ▷       //bullleft //bullbot moveto
4e2f73 ▷       //bullleft bullright add 2 div bulltop lineto
5e2Øc7 ▷       //bullright //bullbot lineto
f88a27 ▷       closepath fill
a2Øf1d ▪▪} bind def
897bcb ▪▪% Create a custom tab character.
3c854d ▪▪/tabsym {
8a43d3 ▷       pop
Ø6aa51 ▷       //Charwidth Ø % width, bounding box follows
dØ4da3 ▷       //tableft //tablinewidth sub //tabbot //tablinewidth sub
c8268c ▷       //tabright //tablinewidth add //tabtop //tablinewidth add
Ø74b1Ø ▷       setcachedevice
38dd88 ▷       //tablinewidth setlinewidth
c69ba3 ▷       true setstrokeadjust
7ecb45 ▷       Ø setlinejoin
ac6e3a ▷       //tableft //tabbot moveto
1c6871 ▷       //tabright //tabtop //tabbot add 2 div lineto
bebca2 ▷       //tableft //tabtop lineto
ea6c7f ▷       closepath stroke
5dØf1d ▪▪} bind def
f5bbØe ▪▪/orsym {
ef43d3 ▷       pop
58aa51 ▷       //Charwidth Ø % width, bounding box follows
383daa ▷       //orleft //orbot //orright //ortop
bd4b1Ø ▷       setcachedevice
131Ø1e ▷       //orleft //orbot moveto
d12f88 ▷       //orleft //orbbot lineto
ca4cc3 ▷       //orright //orbbot lineto
9d3431 ▷       //orright //orbot lineto
59bb3f ▷       closepath
95abc9 ▷       //orleft //ortop moveto
4c945f ▷       //orleft //orbtop lineto
7Øf714 ▷       //orright //orbtop lineto
1e8fe6 ▷       //orright //ortop lineto
278a27 ▷       closepath fill
e5Øf1d ▪▪} bind def
828d9c ▪▪/CharStrings currentdict end def
284874 ##endif
baaf5a
19ca89 ▪▪% Create a new dict to be the /Metrics values
1e93c6 ▪▪CharStrings dup length dict
ee9ac1 ▪▪% Now fill in the metrics dict with the desired width
d4fcaØ ▪▪begin { pop Charwidth def } forall /Metrics currentdict end def
5efb1d ▪▪% End of definitions
157e71 ▪▪currentdict end▪
328183 ▪▪% Define the font
69f2e6 ▪▪definefont pop
a42cbc } bind def
eaaf5a
11e6aØ % Check PostScript language level.
ef67Ø4 /gs_languagelevel /languagelevel where { pop languagelevel } { 1 } ifelse def
91af5a
87b96b %%EndResource
3889e4 ##include "charmap.ps"
1f28bd ${includeFontComments}
6832c7 %%EndProlog
```

```
eØaf5a
1aaf5a
acaacf %%BeginSetup
dbaf5a
db7dc5 /MyFont Latin1-vec /${font} MF2
94e86Ø /#copies 1 def
fdaf5a
e8e1Øf % Compute the width of the /Foot string, by defining P to
45aae6 % add up the x-width of the characters.
7a9f1b /P { findfont 9 scalefont setfont stringwidth pop add } def
a2ae64 /FootWidth Ø Foot def
49cØ1a % Redefine P to print, as usual
bad5d6 /P { findfont 9 scalefont setfont show } def
5f2831 %%BeginResource: procset foo Ø Ø
fØb8ca % This is an example
1ab96b %%EndResource
136Ø6d %%EndSetup
```

```
95823f %%BeginResource: procset Latin1-vec Ø Ø
c3b573 /Latin1-vec [
591478 /.notdef▷       /.notdef▷       /.notdef▷       /.notdef
ed656e /.notdef▷       /.notdef▷       /.notdef▷       /.notdef▷
26656e /.notdef▷       /.notdef▷       /.notdef▷       /.notdef▷
6c656e /.notdef▷       /.notdef▷       /.notdef▷       /.notdef▷
1c656e /.notdef▷       /.notdef▷       /.notdef▷       /.notdef▷
39656e /.notdef▷       /.notdef▷       /.notdef▷       /.notdef▷
c8656e /.notdef▷       /.notdef▷       /.notdef▷       /.notdef▷
6c656e /.notdef▷       /.notdef▷       /.notdef▷       /.notdef▷
82816Ø /space▷ ▷       /exclam▷▷       /quotedbl▷      /numbersign▷
1cb99c /dollar▷▷       /percent▷       /ampersand▷     /${rightQuoteGlyph}
ccfebf /parenleft▷     /parenright▷    /asterisk▷      /plus▷
3Ø2bb /comma▷ ▷        /hyphen▷▷       /period▷▷       /slash▷
Øb8adf /${zeroGlyph}▷  /one▷   ▷       /two▷   ▷       /three▷
Ød3135 /four▷  ▷       /five▷  ▷       /six▷   ▷       /seven▷
6b1ee7 /eight▷ ▷       /nine▷  ▷       /colon▷ ▷       /semicolon▷
b476fØ /less▷  ▷       /equal▷ ▷       /greater▷       /question▷
b23a7d /at▷    ▷       /A▷     ▷       /B▷     ▷       /C▷     ▷
6ecaad /D▷     ▷       /E▷     ▷       /F▷     ▷       /G▷     ▷
f1d4b9 /H▷     ▷       /I▷     ▷       /J▷     ▷       /K▷     ▷
e6deb5 /L▷     ▷       /M▷     ▷       /N▷     ▷       /O▷     ▷
e5e891 /P▷     ▷       /Q▷     ▷       /R▷     ▷       /S▷     ▷
a8e29d /T▷     ▷       /U▷     ▷       /V▷     ▷       /W▷     ▷
Ø337ed /X▷     ▷       /Y▷     ▷       /Z▷     ▷       /bracketleft▷   ▷
9db698 /backslash▷     /bracketright▷  /asciicircum▷   /${underscoreGlyph}
9abc46 /${leftQuoteGlyph} /a▷  ▷       /b▷     ▷       /c▷     ▷
449acd /d▷     ▷       /e▷     ▷       /f▷     ▷       /g▷     ▷
f584d9 /h▷     ▷       /i▷     ▷       /j▷     ▷       /k▷     ▷
7e8ed5 /l▷     ▷       /m▷     ▷       /n▷     ▷       /o▷     ▷
76b8f1 /p▷     ▷       /q▷     ▷       /r▷     ▷       /s▷     ▷
32b2fd /t▷     ▷       /u▷     ▷       /v▷     ▷       /w▷     ▷
4132cf /x▷     ▷       /y▷     ▷       /z▷     ▷       /braceleft▷     ▷
eØ6549 /${barGlyph}▷   /braceright▷    /tilde▷ ▷       /.notdef
a9656e /.notdef▷       /.notdef▷       /.notdef▷       /.notdef▷
e3656e /.notdef▷       /.notdef▷       /.notdef▷       /.notdef▷
f4656e /.notdef▷       /.notdef▷       /.notdef▷       /.notdef▷
Ø7656e /.notdef▷       /.notdef▷       /.notdef▷       /.notdef▷
1f656e /.notdef▷       /.notdef▷       /.notdef▷       /.notdef▷
32656e /.notdef▷       /.notdef▷       /.notdef▷       /.notdef▷
2c656e /.notdef▷       /.notdef▷       /.notdef▷       /.notdef▷
92656e /.notdef▷       /.notdef▷       /.notdef▷       /.notdef▷
9f283Ø /space▷ ▷       /exclamdown▷    /cent▷  ▷       /sterling▷
19cadØ /${tabGlyph}▷   /yen▷   ▷       /brokenbar▷     /section▷
7cf93Ø /dieresis▷      /copyright▷     /ordfeminine▷   /guillemotleft▷
ff02ea /logicalnot▷    /hyphen▷▷       /registered▷    /macron▷
f2c2db /degree▷▷       /plusminus▷     /twosuperior▷   /threesuperior
7be9a5 /acute▷ ▷       /mu▷    ▷       /${pilcrowGlyph} /${bulletGlyph}
8e1ffØ /cedilla▷       /dotlessi▷      /ordmasculine▷  /guillemotright▷
644aØf /onequarter▷    /onehalf▷       /threequarters▷ /questiondown▷
19e7ea /Agrave▷▷       /Aacute▷▷       /Acircumflex▷   /Atilde▷
a11d92 /Adieresis▷     /Aring▷ ▷       /AE▷    ▷       /Ccedilla▷
763Ø5e /Egrave▷▷       /Eacute▷▷       /Ecircumflex▷   /Edieresis▷
6d3b5Ø /Igrave▷▷       /Iacute▷▷       /Icircumflex▷   /Idieresis▷
d9a88d /Eth▷   ▷       /Ntilde▷▷       /Ograve▷▷       /Oacute▷
524e4Ø /Ocircumflex▷   /Otilde▷▷       /Odieresis▷     /multiply▷
f6a29e /Oslash▷▷       /Ugrave▷▷       /Uacute▷▷       /Ucircumflex▷
8b3a86 /Udieresis▷     /Yacute▷▷       /Thorn▷ ▷       /germandbls▷
a41298 /agrave▷▷       /aacute▷▷       /acircumflex▷   /atilde▷
e9f1a4 /adieresis▷     /aring▷ ▷       /ae▷    ▷       /ccedilla▷
a6Ø78e /egrave▷▷       /eacute▷▷       /ecircumflex▷   /edieresis▷
d5Øc8Ø /igrave▷▷       /iacute▷▷       /icircumflex▷   /idieresis▷
bb5dff /eth▷   ▷       /ntilde▷▷       /ograve▷▷       /oacute▷
869658 /ocircumflex▷   /otilde▷▷       /odieresis▷     /divide▷
46b899 /oslash▷▷       /ugrave▷▷       /uacute▷▷       /ucircumflex▷
f3Ø29f /udieresis▷     /yacute▷▷       /thorn▷ ▷       /ydieresis▷
294Ø31 ]def
86b96b %%EndResource
```

Directory

books/example/

```
--1889 ØØ1b729f8dc8ØØ2ØØ17 Page 1 of Makefile

317ddØ BOOKROOT = ..
77Ø1a3 TOOLSDIR = $(BOOKROOT)/tools/bin
6bdac9 PSDIR ▴▴▴= $(BOOKROOT)/ps
b1af5a
e3Ø1Ø5 YAPP ▴▴▴▴▴▴▴▴= $(TOOLSDIR)/yapp
6b9b1e MAKEMANIFEST = $(TOOLSDIR)/makemanifest
234d56 PSGEN ▴▴▴▴▴▴▴= env BOOKROOT=$(BOOKROOT) $(TOOLSDIR)/psgen
bb9df3 INCLUDERES ▴▴= (cd $(PSDIR); includeres)
dfaf5a
ea54aØ code.pdf: code.ps
cd8539 ▷       ps2pdf code.ps
1eaf5a
ba1aa9 code.ps pagenums: filelist footer.ps MANIFEST books
b179b8 ▷       $(PSGEN) -P2 -l3 -DfooterFile=footer.ps filelist pagenums 1 \
1e41c4 ▴▴¦ $(INCLUDERES) > code.ps
adaf5a
693Ø8c books:
81c7fa ▷       ln -s $(BOOKROOT) books
f6af5a
5b2597 MANIFEST: filelist
365dbb ▷       $(MAKEMANIFEST) $< > $@
Ø6af5a
9d7afc gv%: %.ps
c3c8e7 ▷       gv $<
52af5a
3849c8 clean:
bfbd2Ø ▷       rm -f MANIFEST books pagenums code.ps
99af5a
5f1ed2 cleaner:
b796dd ▷       rm -f `cat .gitignore`
```

```
--8862 ØØ14f236Øc58ØØ2ØØ18 Page 1 of .gitignore

2437Ø6 pagenums
739c6a MANIFEST
212Ø65 code.ps
4f5eb4 code.pdf
```

```
--1edb ØØ1293863538ØØ2ØØ19 Page 1 of filelist

9cbc4Ø V 1 8
afd2ca T MANIFEST
67b4d3 D books/
66b2fd D books/tools/
2d9c4c T books/tools/bootstrap.pl
6c8e39 T books/tools/bootstrap2.pl
77c8Ø5 T4 books/tools/sortpages.pl
2ebØ49 T books/tools/Makefile
Ø4dØ7e T books/tools/heap.c
7f34d6 T books/tools/heap.h
f338d2 T books/tools/mempool.c
a4dc7a T books/tools/mempool.h
2cØ3d6 T books/tools/util.c
21e77e T books/tools/util.h
Ø565a3 T books/tools/repair.c
722fe3 T books/tools/subst.c
61cb4b T books/tools/subst.h
ac7daf T books/tools/unmunge.c
bØ3328 T books/tools/munge.c
cØb944 T books/tools/yapp.doc
cf915e T4 books/tools/yapp.pl
b6f933 T4 books/tools/psgen.pl
de1aØa T4 books/tools/makemanifest.pl
f3db93 D books/ps/
5c32bb T books/ps/prolog.ps
abfØ24 T books/ps/charmap.ps
5dc361 D books/example/
baØe8Ø T books/example/Makefile
d2Ø5bc T books/example/.gitignore
82b563 T books/example/filelist
2c365a T books/example/footer.ps
29854b B books/example/us-constitution.gz
```

```
--1fØ5 ØØ147Ø8Ø8528ØØ2ØØ1a Page 1 of footer.ps

b5cab2 % A program to print the page footer, using the magic P function,
9cc889 % which takes a string and a font.
c84546 (Tools for Publishing Source Code via OCR ) /Beteckna P
da448d (\343) /Symbol P▷       % Copyright symbol
479Ø6 ( 1997 Pretty Good Privacy, Inc.) /Beteckna P
```

```
IK>H&P H+5ICD^iZnMCA*X#LYN9duN$cZS%hGq9dkDhjX8V+#cT:n6?Bh@YntBZ8N6&&?%//4u5VD6&bl*VZw@Q
mPD*1E p^eTwSJ6m6TRXKuqZ*@!CmSATqZ6*b&#eW:5mZGZTBBIJQL/biL:jcl?Q@VV*M/u8S*/c6lXM@#6Q@y&
A%+APP Dp*%j9b#tQCQd\>ldLji\j@:dk#?5PtCe*<QNV%>DMP^C/99%qWLk+=NGK6=1HdDiKl<hSlGjGueli*5
e%j/Te Vn%<6/:CNQ6N%$::/8eVVGed<M*yBA<&\!HkZ/y=>Q$@Vt*Z#yi/Pw:Da>5EHBK@/4dD+RD#d<I8>FPk
/i*TcF *\@DHGlGD/EQw??1e?l&jwRjjh>WW>nm/Qk:LGcc&9=le4CE46&Eed:khIjVPIZ8ETmAL<C@18M\#WAI
8yPG$J uEA#+Lw<LA>@nNN*<:/hey4KFK<T>R/y?ucEJRIS*#u*H5mC::&E9*bh%aYHh84<*D+C$ab#GKh1H<aA
5mQ9kp H*+Ak8CI$>:De=6R\XV^*#kuuMLDCP8CD:tXdVu9&?%i*6#CdN6y^mu=/u*w+Rqc%J?H$+Vd/9+YnnBW
:bHX/b /Gic^aRV%pt?1y#:HwdZb/9m&VR=&CEkH$\BHn1E\\aZ56*DZ@w1QId^+tBHB9n1Jjm^&q^YPG9C=&Aj
%>Hu$+ hR$+S@^:/GA!RPT&h<HD18l*?jDiQ@It@?Qp8*aH=MbyV>WakJWtIAq6n+\\!bjeAjc=/p?\Q:C5ckN$
Z8pFPy ltFqFn#/9<t8AA<BPj+/ID>QRI?+ZF>RAmk?!K@>MQP<@D\DA\XcljMQ9NA#9HF=5eNZpBT!#p$APmEB
>huN%S i9+4nL>6=@dZWJ^bb+tVjSL*HGtNT\%NVZQ5kYb:9&quaCRi^<<<AnIS<<k?tD&p@aIFq:8MZ&9cEac8
#VVyXR b9@Bd45@HweScR:L>HBEH&m^?XdKq\:?kt?$@da:NjD+8aZD%Fh$w9/D^aVjd8HW6VnKkG#hn4CV$^Gb
BKNa$% EikFP+b5qZM*nNkEPLWHwnR\qt>+AJkGEKnV%*D>8#EE&WpTZVkSjldYZY&GE*Ie$DTMn!CmmHY<\:H^
VwRECq bdNd+aP??c=9Yk@<lu8PqGqRbMJZaIN*<=+aDmdyBynN1F??1!w<Wu+:A<>YkjmR&Dc6/PBB>pZ/\HlP
=RV*RH 8GI*B=a&8*aEHn/EMPM$lEiMWFmS%a=+lkiqI%qNy<$EB\Z&8hJX#51BJN1uE?%dD\K&<lPPCiiaJ*ty
?Y6E9! Pu<YyyAXJWK9T\ppEyA8AB5D9@DJyHuAqEkt:?E+\mTBiW@\=Pyb>#NIVI@T4A<Ft>:V*y#/8i%++qhJ
iQ%PY$ Y=8VD9L94?Fe@hpjnlNM\=\GGLih8$A5ATeDYaV&w1:?+1eJbkemeb/uK4RQ?:q&j>5Q5Gu9RG4ALQD@
*=N?ZH j<e/4AhP?k:?PI6wkwuZ8FM>e9WYFw1!wK=kE8:@DPM9%hS?wH>X*\K:lA8R/VVY%Zil:/hL>?n:%55Z
:qC#bh 1#<E&ALRiD#pZ@^+\Y<:G8CHhu^w<XuaWBkP#8m??4$jl/:69Z#^@ZBc@lEH?FMa+GZQ1\Xy<HeWLI>?
9+/R!A C&K!le*e\=pQKF1Im1QbA>TGwJllpA6Vq\n6@9kBqDGh+Lj\^REXLQ=>DMAyVP%&tp:A<QAL<T+JMS\N
QmGl9$ na1&+?%$eRb*#>JJee6Rm4\@B9<LMnF1i9\&&AKkLaDC#DYFNyqRdyA&^9Hd\^Gb<RBj5L4^e=/GcNSX
hSG=$& $&yXP<:ukX>CnaV$Mt16Xy$yG6LwITXbVReHNyuQKe<jS*A!1F$Mu4NHMJSk*8<Pl\DCck:hJetSm=Hp
CS@*d1 $^VR<CMDA&$Gpk#d8DKTPE/*ym:idq4p>RcqSEBhnB?bp5mHnYR=68*\P!wDy?DuKK$mVi\HK*wljwia
@Z5Nep AApMewB5k\@HXp=+6CKmtXEZj?PlRAY89+pHDR+E9$tM\u>H/bSiJw=i?Y#:Id+:DcSypjk9?GYl\#X:
YK=TAR mQi\*T$QGdAB5Hd:$H5tP8\85i?=BT9aNchSXB$<6VYC*:IN11D5QFd^QDw>lVm@A^\RpBdAy!NIEqL5
G<%<8= 5G@I>JMIXZ!*D^>K6iQM%*LhQKQkmAi+5kh+%cSTB4=yT\$$D#cuuGmP$yL1+hHdlyT%M6%9M1SEj?^w
A9t\ZV 94Bt6&yJIbj@AILt^Zi+n$*R>/>Km9I=Q6YBhcI=&j%AB:8G9WH@lw<ncm$=jZIk:#Il6TIb&VSScek$
^68W!e >B&\5E>VeV?B9@WV<$d+1XdS:<F/MAICq=uiIX+qTmyh+NTEeh9!q?YL?8=6/1\6ap%YI&<k//JGTVId
jXJXQR lR&1MFCMM:E6i&!\k<jVmFK=qGX+1n&ebnV@aJdnkR/jn9L$hEGh\V^cNkdBK68*aXh>!*/9pPLIA*^D
dydnWl =P/Zb\FEA=mk5%nJwKlyJQk4iLYnEN&BItH=E5?>Pq/mD+F8*=/cl+BWIKu8J*bNDZW!&\!MlcdcwSkp
6VhV6k >iBSA:5^HIbH5FSJ6jP&%iISVJudljRHm=86u>D@QF^I<9A4/H#9l*+A?VmQRA9mD&C4ZV5DC#hl^au?
/I\ShP lumiHEWQy>\qw5meWbDC#<V@RLT8BTihjX8\dH<td&YAhVj\Em/%h4A49184u!NS<Nun1RjQul:IHBA$
<aKiu? eK&?@+K=eG+QKDiPuh?Kb!RGMm&\w=hXT$:Ec%G^wc\ynJW/:Vu?jV8&G>dA^9^h&kI18P<nlcLyn/Le
jQ=\8L #$S+uKpc/mVhpcE:%Q^pV+*l+hT^WX1VZZ/e!eluAT94k$YS8MBX/V4pPMbQ<k4Wj4At?yaeVVd!p%/q
MMdYqQ 4/5P5ce8cj&@kG1EleDLj\DFlp^hCDycqSMqBdK1cZe:IAu<N:^uh?%L=@/IyGS4MZ+NQ$X5wEeudqpN
^A/6DA 8lSw*QN&FZEFQD6w5b^:QY5mMaHAQ:Nqm@c4X^YIQM8lI6S9@#kN+?tAFG+G&DpkKm!I>>BZ$\Gil:$S
@G=CZX L+4&5jpbdJLi$eqDi8Cd@qIL%=G<>&^T@!y?C6mbkVW%cE\+#X=pBeu1PW$/h?=1eb$c+R&RS#%\Clwm
hk&X:d cNBF?nu<1MVt:nLC+k@SD&iL=NTFTDB+B@$?nMLXe8D\:$+FFl+VECeWT8^\ZTDi&+HuXbMM9C$ApaF^
/MRNaa +X=YiLi!/:?EeIRHmKRk#6btlqRka6QS\SkE@8@^^KFL$nd?EQT$EQ5T6WA<VY495uIE#S1qnKNw%El&
uc9Tn/ KNlyq@5a&wI#\TKp>=iFEEhGcppTM?&Hup+%YPXVetq46aRdWCl5:cHbTiEHHlBX9akeNGae@C<eqEdQ
SAQSck t=ADVVL:6dw!^8YEAXT&hjZR%@y=w=?I&a%TJNPmXm/tQcdaMC5YhAiRt&GwjmDhDc9VTApKH**P#!qc
9+m=W9 Sja+?E<$S&M%mY1*B@?%MSyE?F:^FNF&S8@W<@HyJ/?BZaqBTWAIbk1#JFeJdCY^S%pTMi5eja8B=EmA
paN%Pc :$ja*^Qa?16GDdqAcubkWBhNM>VRWS4Q?D/c@d9RDWAe5t>HPw+!I14^WLQ&AIP9C\>BPNnDMRFBtuc^
=6dkM* Hb$SCeI%bteZ+VctMc<M\>GD^PZ\*!\m!bmm+!Iuk#td+tdA1VBT&JIb%aTNBB&b^=#\tLG:$ADblMnh
V5tG?c E8hG<!F^BQ5N/mZA@^W^YGFJ\:q$j1PJn1dw&W<85?%bldD+^P>6pa=1umdL^MjRlhnqN!L8bJ8&@Sc*
?ZS9@$ D^/S?bCw\Auhm>Zu^PyV1!EbT>e5w6y/T!\uZCEt>G5aXaZ#CKjm=EWQ<>n&nH#FPakN8Gk5k$jjp1wj
t8#WLw MSS>W5pJ/*=wi>uB&aSiTt5t&TNm1+8?RC5d>+9qN<6#aJi?Ij9e?aq5dqwE#RZFN^pN?K<qca>wc#Vb
y:jt*y VK1XpLXLE+umclN&KIJcbDa+:WlHP4/dKCTnn/qG?lW4@^+h>n$^EClE?EL$@>Bjl!?R!?dkIZI1+/><
m>\KR\ hPl*wkN8%EDYKq5j5$RQ>4T!6ui<@Dd#@8KPwHcM*m8Jhlc>eWCMh*S?jFADC^WVm*!SYc#@<LNnwFRS
ICpGYE eq/LMh%*P\pWmdtw9$w5A8Y\hV@muMWijEABD<QmdlE>eVQ46BAW&/TW=8El$N=HQd?kl!kV6&i=AWK*
>YCh:m wE4Z@lI\KIKnyypmiTT$I68EAu9lPPuM5<pq:kBT@P+$1+kta4AuljNCJ=!G&R!+taXi!qMVtd4uiD#F
Rh1$H^ XPAwVEtmW=IA^%XjqR#MB%A*E=<6PlR^I4c8#/Mb!q8T^Hy*pb*6L&akJ?KZtbNuT:PJS$C#EiDRw>K8
4*Hid9 Quq@/8p@YFiF=qQH#JI#KwXn5yP%d\>%XBlYu6E=BH:H4VSJB#=qi>@R8WK>MF6jILplQGXWe&^\pIN*
4BaMwt JEp&p/uHm/8VNaG:tPa4JQDm>:EAIw5WZaJy/E+^B9hI8ijBGCY=bYwCN%t1hTGq^In9KZ1HadhQSQYn
>HjY+h 5iim1ae^9LTwV*Pj$e151ySb4yBbHCLdDbIdN*5\w+eahym%V&4GtYQDP+hG?Takc+K>99k%Vt\ML$SK
kTYM1M 4=!*WD&DN565wLkW?FLZHqyN1PS?M&&G=^E>edlXiF?981+^qpSlJPjI>:@Rl:bD%<Y$J=wY/BVXGEAb
nR$>a< yuuTdGA:mZF+>MTEDk\i?dXAMqR=pwEiKRAMX*cP9ETa:I$yJQKDl+XwDKDcPJ\iakAyH^JwAid5MI:D
d?GBu: GR##Hp\^pTFk8DV6Fc%HL$uwkFeV^m$uK*GNAZXj4%EFpq!cFhaE9=65Dq6FQthTMY/&$c<p$An<IC5m
lYjc4X H5Cb>SkIAh@<X>FG:MD!M*CA$cIku6FIEEyw*@<!uw+#Bhmjy5\Y:EKtT=$YVCLjh!iVnyFWkT+iZ?P\
+Z5#CK <1:>\P8i+X>?yKjPqL8V8BF=B@1=pqFTYI1CMMMJ=j\PVL*pqaHEchqJl/?^?Jq=&6i*LlEX&ieF&+VC
Bm&9uY 19mHejBNtqEE&eB@^>V!BARpH\JJQwZhCbR>PMCpquqT4LkeuXKWVSk!I@8cnPlJZAwW#Qk9V5h58>PE
cp^SD$ S9!IBBJ^B5LePCTaH$R@ZaPQY^A>a=8&K$bcpIqPwASBijR:iIVhT?aGpBaH=N9VPQN:eBI^bJEmkSZt
Gnaq1F h4bCXuc><B@I/8BqJp&:l+qmBp5$PXATV@TW1YwM9VG#+wHRD&+:QPbunIeS!Kj1Ak#ML9@Bn!D&K*c!
5>*jG> \Ktd%FCnCB=JRQT#\REq>&94n^kRJAM/by:Q5N&nuFVjVlwTJckGdNYB=AjKp*=k8\m?E:NbLAuWC5ZA
>Bqaba R=R1bI^WRI<HNDMSMeA8PlmCuI9MGJpuDd+XwcmABw<ZjmC6JDL:A&=qH$P!KVTyuB#ZB5PACIH*6t5H
pd6FAF b\5#Fi+V8ZJASBpt5+H%Ki<XhGMmHEk\cpkkauFCyw*Sj8chCPmnQ@wy:cQ>:nA??L:>FV=hE+HJt&Mt
m:$qR5 da<\I#&#Ml8SD@AmL#!LJMt$DmcBGk:Hd4kpBceIqZZ\F&yS?J&D&CKA$Em%SmBu=p&TL\b65p#R$9MF
wWdmkc lmyZ+:KwPRYnhN%bXB9Ek!yipWYcEE1uQ<Llln/LmaCy=ihyRA$CN5mnPmZ\IVn5IN+@TiATtkhauVA:
GdE^l* XIu@PMFl?%TITVJ1G+MiD8uhdwHZ^tw1aGi\KDDEJH=1+q:mGe!T96a>MB^<NhEwVMl4JnM*$6dStuC=
yeQu:m Euk@u5d6*p@JFC?Cc%&i+mZWXa:5Y6nahV^8Rdld>BdIdueA5jW4B1TSkTeM@/K=M&^@iZH^FGCN$At\
*VW@M* XpPDRu>#cmq\AedJ&:/y$>ptl\\QYAHWTtc8YEdk:Mm^HeeL:l5mBS!j$%aVA1BZPjdE<ElNWZacGYLR
tmNC*> yIAnmXn6qPL8\pnQ&mICGXIa5kuIGj^w#I/1!4eG=S5&KA@6k?IBS4LM^ntKL\tw5%6HD8Aqwkd+aQdD
```

```
&C1h&H GSaN4J\TI8l5@cIpR5*bHNywq+8uA4@8&$$dq^%e6G8QX8pS=QLq%=>>ZW81Aa^bm51h=p8EcCk?yZe$
H#:Z!a Z:MGC6Dc\@Z!M&CM49MT6!e*p>eutcEam>S%w8dG@@=y:#ikMH<CeZE<VV@q!=qPIJ:/Bu^&J?+hya>6
c=d/C: *uIMt\ACZ5k5+qm5JVB165D98<CLmpH*h8c\IVt:/BRRpkCDHDDGV?HDV&n&@LV#&*u+mjS%e1:IkiG*
1<j54q Re?K\=FAGAaTtX:1MAa5<&>:Du+SRNLJdt#BTnlNPInFy\utX%\#Hym>Pp^Y\cH1iS=&/@J9w5jIKGQ$
>TmSSd !jGnMSTM5XnpNMFhnIFTbX9njV?$n+\&\Yh#:S6NjA\QeqlRq+IjtTVq\F>k$e^JB\KiEKH8KeLQH+N#
@Yj%y& PwX8ew1&JHJ!cqpHwYneAQ9=\BTkCEhh\ewI$#PTq<X8pKQ#Tl@l^dH%p8I9w#@*KWa>T$Ci#5EC=8R!
4#SnqE Y&lmFCdH>?<q!<ipEYWS#SZWj/jpeqM?+SlhM&V1Xw!$Ha6+b6bm4yJtpq%#pwIm+%u&H8GIbE+PG<LJ
qH<L$R t@$V4Lnwm5J%jQ6W8Dtq+qF1aHw?ScDX1/>hAWY\&mmyS%cdmn1Kmch\GLwVJ?EXRwCj\lZX?9A9LaGc
VD1bpN ae/p=TQ9@pJtFi^L#/<=hdYW1SPt@nMm%GSRw^\VJRRc=eTBA^Djn*cmq@F\Md8kke%:>^!d@N/+l@i\
p94L5L Pk\wW^Rh&Inh=*9SlV%yJKA*<6!D\mPt65PVRwMRTJPm4n&!6mkwnmHee@k$S885AK9Z^Be6CHQnIMF>
1PG*>5 G^MlhJ#Mti:BpN9P4QN+9a\aKDn541SuZV&LC#l<e6EklYPS%dKj\41FSTAJ1yV^\6&Qti$N^h#AYHZJ
RCFTHp eCCF\@aBtk5#Yi6H&Z/$PXJ!6R<a@l8cp^#8GD6l>=NuX4Q\FMpuMt^>ml\w*TRQZiJYtYBb#^Tw^WK1
Jhd!8b FYibjC1DXGEe*&NLTw&DV:BNR%iT>@td56\RWl^w5k=>:^?A@Tp4Dq<bn5\M$:+>E6PDi$1GE+wl?Khp
LHyJkq G9Qn=CnI4&aTbp?B1TT>\%JbJj1kYb<aJ!G:na=#/B<R#peukb^mWb?MpP<8yPuSLlADVR\WDK5V*kQh
u>RRCb We\luX5^J5la\pp*wYM$TC4AibiZ+H^9<bACI%568LpC?ZLLiy@1#^Gnl1hhSK^NTk4YHkGuAq*pZMAm
KY\G<e D%JC/kSA1k^I%n&P^wq>#t1mmdpD<&\+V><IC5Km/MZq8\L9#=CncEY^JXKn@^:S@XyPW#QaWFt*4LdR
@MW@^I #L6d!:^u\ZPiu@1/C46u1#*c6Qec5dmc@&ZdwFS!#mANTdM>mV84<@IY!k:DIuVhkq&PSWZ<MPKbE9dD
B%P6F1 SK$5$JV!BmB4N9AG<Cu6+*qIJTq+:QbmCpWaXb/h/F$JSSW?aCI1/^!+6P&PhL>J5:NPey$YlDFK%&ch
6yPwSX RlP6?u9bI*4cDEIJVww=BnlYP\EpaRHjlAeQ*PI8bTVETc:?hMZ:mmXN%8lhBXq!$8CP1H<FJ19!WD\?
*4VlPj =w+EM*%G>S65AkK9qVC<R5S@&hkAJ@$T!NpSbBMn6A%QV:uRQ?1lj$DDI*DhtS\RP<wt=MDTG9NihD:q
M?#<\% iW:=G=Db!yF#pTen*$bwPjt8XAAVIVnIbbk<y9RHk@JpjF+yA/>:PiZF9G6Gh>>CA!m$^SRiCQR+HRGe
\VapRS k<c:MGReCSAuu6C^QJ<1VbL^=Y4ea\GL\/B5IP\lS$tEe?GI6E&GBkMP5auGlIqPW:nV6G>ERBbdXLKC
Nait:^ 8t1a<%K89qana@Ch\mXc1*EjpaKPBZ$:KXJ=4cKVaTAw%TKanmkTe&:PXnc8+lYX&W55inK:EQcDbED<
EL51A1 TQXPIb*wq>itKI$jYGC/c!Nm*HPPlIXpJkp41E*Wma5>W9/i\A41$E4H\DQJXkYhSWauducJl%i^<eyW
<Xuj/q B<P1T+XeAtGZIEm%unIPT9yV=IqDdPhk+ua1cNE6F+%CE%cBnM>uSEENeDQbq=XeKNlZp&maBP!%VwK$
Yh/bml /JQqy*P&QI1ZN5M^Si!LKttVWR:kmLTIE%bkS%VQ=8#*AtpTt\W>Tb:/akBTLnVC^%cN4BjlWlXRMHY+
H^Y9nq C\SEtb4F&1w?=y#=9DNmjqHepEHwB+cMA/WcPAj:&yYXQt*9^e>H*G<:djiE++tJcptEmX<qMYAKBepZ
/cb5&M mkl\E^/H:nEm4W9nWyp^a&iBn!?!&LJcYYu>=BJ1^5e5KIIjLTdmPEGGy?Ye6ui$NWZwbaDBYp!i%KW<
tXZu4y LR+YQcnbCe^/yEwtlC<IiVi!biJm&M1ZXA*K%4We?dTAlMIqV/jS?kAFqTLFeI<Vl6R:C#QNpaQK>^8L
+DkKJ5 H?\JHA$!lcND:Ep+pcu6$PRA^^J<E<GPyNJDPmJPy^kbG$6miMRE9I!Bt/Swm:JmRZ8KFkeC@a/DtQ5k
6+/cPI *1L\Y\44C>54t?+#CA1!IQ59yBTZlVnjXQ@=u?<RRa?XQ+ei4I4ynBH8V?Jnu*Kq?#9&CHei%4e:wTNy
4\+Fd! M+DQKpDK/?%^NJEpt<D/>\B8>Ta&4cDEG:iutNGyS^8bpB/NP9C*i*He@e8@&PaL1:!G9QH:+RB&C#k9
<\t#aH L>4?BheZ!ZB9^\NuX@:1H/>BXW%6w&bH4:SkwAA<=qTBEqLEGR9ZW\=!LiK^b*>Pk@A+m?Hae1IXM%Te
*Z>t$d !w<XtlVec+??m/hhT&DWFwk%m!p%A^M<HRWX4&GWMYDp&tRm4ZVM+RRCjTkPBkNM6b!MR&MW>A8KKwZQ
@=<t!* N%4c5$n/&?S@RZdS8t*9%81\+N#<iVtHtmNVBmjnG/K/lIitlZ6Wa*Cic#uS9KJW9$n9y4wC^ByT!bkA
T8&/bD &i=hIT$PLeNwTW<?h$w8A>>Y5X4u5MEC?wbMIEumTWRebh!5tk$HP*WkacmTIp%bMB!@S$%hbkM%jZ*q
QaldHl GJ<yM=HX6mIWk\\G#5DE<6yE6b$Ww4kmuStlNLEM+Ew>qmlSDCiS&C:T:l#QW1TKTeBL5bQNBSJC5qCl
+E:<ae HhZ@K@aN1WLL%&C\@LAQu4dhH1B%dXdk+YIX@8qIlE=F!4NEc/p5TC1Y4>ap^#4YC=<1R1<kywcT<dDL
Ti//uN q4yWaB&4FEIWYP*IFelT6%CW+<eDbG6+?H@bpl:!9RGcD/JW<l^/al/FqmGt?HAIjSPSTXWGl=batmC/
<Dn5KF MBwbD\?ZtiY=d<\KNT&b8m\y/+45<uu$bK6+LB4bdlq8WqM=cPkD&%Bcq+FBtqcaZlI%W1wl%ltmI%KA
iRji9X hlQte&CQ@TDRe^GGTli%=k*WFJY#M/Xw:BNln^R6b1n5Wm&uGl%AbwdLnu%CE=SH@:/\B:ci+^XeapT\
d$HJh6 Aw#BL=T/YDn1IT6nSijYTAc*=l/Xhb:G8F!#qPp4yqVp15k^pFk!Mc$VIKtGYI5dJa&W8nJn%1pRcINI
y=9?a% t9Fme6CER68PlLDDN&HbDY!+::qR<*dpG%m#*dckhYP9m%9B$BRP^?a4GbR85q+8!$Ra?YplE?6qp4Ni
eIWyeY :dkq$XbYAdpZIVN\@Pu5p:YN\E8l8!AhSWA>DEPnhacupRQ\#:Hq<+1\wta5F&IT5q&a:pq?*K:PidY/
S<HF=y TTwkb^FGyt!5ySRa$\@5L4BJHjlSFHjRWDhZBSt@9ADD4dXHWENCi8hEb!N9X^TD/dC$CG8/q^tK!iMJ
F9=j=p Zp=^?a?#Ai+yAlZBaCTpDm&a19NFeIbwWB/RYLDklw@PQXjEak+N#N1JNNdkW64lTkeG#?ZtK&>HEZwi
\=n+\M uMcXFDct9*qJ#NHpPaa!GTlERnnR+<+?pt1a!$H6W<tyy=G\w$yLkEPETE/!DcSl4\#Kwa=568VEpjYk
d%??5c H<4R#Xt?HhXSuM/wl4hERhIYjjSYP#Z*JBA^VAdL=4Pej61pPlMEhQbZ!1T/nC&\@5YwBbwR#hp^?8aF
WI!X8Q mtI1Rc!ZtPG5F^WaQB&l\8*\6TMb^bZ$il\^Q!>Kq%6VeRF^BMtu$t6dq4<RuCtHI=adNVVREC=4AiNw
qAjCQj 4Db<MAJ>e\HViyS/bNdwNF=<aB?/lDuq@BKFYYNEuV@y!ylD^E1#RW^8JE4DlKGYtENM86h8n56Jt#\L
S>F<LY iI:Kkae^Nq&a46uan8cktcT!Y^N4$i5?Ylqi:!u*ZHZlpeCJb#</<?+lAkadIE4!PX6EhkWGYaQbKRL@
X:?1/H WCG>tM<QmLpEN9#44eajqK@pCW5?uPMwDpQt:YGRpM6=RFymI+ERi#iRiECyleE9:4R^>I^!ab5IjAnK
%#H^Q$ elFL*%&9mau&9&LYq+WJcyw+^bMuiF%cAA>=^9bt&&HEGk\^8Veh1GHF>GB^A@WREka&4pVZ:pJiITej
>#@VeN pq%PX5+QtbH#4!eNALFA=5KYA9ctwRdteMY&5akmpA#XQ1LVnGMphb\/B%BYuP9FpyCHDZ=:JcT1FV?p
:/Nwk5 iCNCtmM!uI?Kpw<P\HIY>8PtGw^ntLF#J5ukNhm1tBbY4^kT4LdAI?CcAn6ba4iHA@CCBqnmaBGp#5Gp
>AbZDF TT*56:y%HMmPFh/V!X#uFi@G#mXJW=6YVd58lNSNqyWcS8>:GVhV6&LbDV%+9\eV@+ylyWX>6<!KnQWa
e!EH*P Y+X9W6W+nNLBT%l&iVy1X\C%:!*wZuN:XpGw1$p4!9I!JG=NA$yCet$RJeI&6/+t+SaIT^S%NjCSQ!1M
*Ej1q5 YpqRt=^+$haNmW<XK1QSP5\8YMP$CTu4IG+LeiNmiq#ydIyiE*FI<KCFEp?9q@c:BHRw^uP!ZIGCNKhR
bhK/e$ mHD5m!mkcWYJNcJSBQywuH9cmBbNInIGSWD%+i\KN5ewAE4epaH^6nhyYG^hDI$:&a%i&^%\5=++uE#w
H8eBip =%Pj66$pJMYa8wY1NBn@1aCMT\pRujdJuRM%+FpBmn$6&h9yGNG/VeL9K<mjK+:!wy^WTyIRVZB$B11Z
P^D$K! XSPQkkCWZMGCIeyaDNi\eemy#LhLF<XWm\<M/^>QnHKcCe6M<yl^q5Sw$*X\Q?@iw8ln^jqjM\5iVy=R
j+6!6t 61RmyIH1HJ5mKJ>ljteM4D%<##&^+d8y>5CGKVSQaCLhB#XQuPLdtWTw<a:HVq66LVqEX=lH!#<=K$4D
NYBpAp Fw#A#MR=qelT<&ph#cLapS5<+\=FScbkj<l=muMEkXDjYZB=YR:jBFleP!e\b=&wRC!aPpJWiuS#Z*ZR
q8M9m* ZJK:VayJ#qwS/<+XWm$Cc$8lcTFE:PnNe&L*SYeq1d4c>DFlBETKpQm8newTnbNi1+Ab9<FB*XVpXST1
EpL6mJ TMIFI+1:NptRySmkD5QW4nKG5+4FIc&\%Ct8T6T!jVQ=iN!jkdA+w@GKD6nSF4P<jp=N*pV8!1a6=P/>
5#<K:Y p:/*=D%=6ktXlHpucAwdZKqWecp6Vu?JGMXw#$mdMHbP1Y$uuH*mNbHcQXB@n#$&5\/j@BJ#5kKFiawm
$8Ck^u !%WaI:BalHF6FS5cRG!d+^MMnJyP8>$wAFaDW:Nu=l**YiG1eAXjB&&Wl?lZbu96=aC/1:b%?k18eA18
uEJDB# &K/CK*AutRi9?nn8cmVIV<c\!CD^yEiY#+8XiVX/XITJHP$*a6^Q+<Sl4>nGVmP1V+YcHQ@>iq8ic$Eh
:Q%6#M G!cRwq\6K$&Ieep#yj:a8bpkJT5+=\&G%mK!Q/bidHy*nISJ^wGYABEHnBj58wEeS\Zm*L/K8?KcqqXn
XIdVB4 kcqd^ml$>1q9&FIkbDIpBdCN?Y5<G=hTWDdetqQYGNGq/IqAiTaaCImXLH:NYRT8TX=\w%y4wEwS:aN6
$/R?4@ ?kpyJ5HQS+bITHcbnAFjLZyS&8cVMibWXLe5$\Dy$L:n:empu!#b!qW5q&n:pC<jiSWFR+8%<uM19NT\
I$8FEQ h5LBq8cP!hNIy\S5KE!V8MVTj*>k4i#iym=Kn1j:Qd#::^4*Tn>@R&H$*$pN!h?wC<Gh?$=:hCRn<V%&
```

```
Ay5ACL e!E<*159/?4qX%?@8ZN@W+&p*Sm4q@Vdt=h*hk=+9eXK4A4qK=+=a#Rm14RplY\%/Ztu6a&/at4hd^9I
LkaI&\ 56^Au61MdjNku&5w9?K+@bk%m+/MI=i>+EaDMclAX%FRqRNMm^GQ$6E!%8+iA?!TXl\cQDQZ^RMncwiE
6V>uNW B+RLwITdC91kI5A?H>$L:TnSyJy\d16mRbi&#\MnNYtB#nS#Zq5F9MnND5pyByZ1^<PWF+?ChduD/A1H
Xadm/% V?\M**F%CiLuNh6RL\yB8FWdcFJSX*VYyIiT/FC*+6AMb&w&&R>bcRuXjElFuqhhV#/%!/9HyTG4tc%a
M$Gbeq w%iI5Aq/D$\E=piw@X/!1EiQj?L<D<hLytRKu*1PS%Bj&Rpw8h&L/J6C?!9L$w!1cDL:tI5E9wi%eTni
J8^uGV NL4u^SJ$Bhl<R+N9$Z1yuRnEeF%1ey>BhYC\$VNVTP?<pNcJ5umc5hESm9NuZDEGn@SJ:CCcE@iH%T:h
RcH!&L G1Q\@ZTY/t$9%dFh*y<ee*L?AGVtTEh6\yli+TbcPyTc>DKZ@&uEGZ>:RmiBw+E8l4M%yTXcq#yD\BDW
#YL#k< GKLm^lAY>^CQF>4YXC?c+=h!S1!Ty8DCIq8:T5/n+qAn=Cm4^lFApBd8u9ELS&d6u+E9@hlR?SZHM$j5
lKJPGT a:T^Z=C4ApLZyYPF%CNlVtq>/WDyJPkdYZZja5uHnk>Zac9m%&1>?pnF/=H@wSb9e\weXJ\WQGLulYPl
>>j^9i GYG<i5hR>%$mGIL4hVkM#GYy$aK**1X$pT+Q^y+EITnIdL@hu1da:yp:Tmibl$^IR4/SjkK1+KuHM+GW
Tn9q!: MP?Bb+W#RJj<nl<C=&u@mLQAj:n>WZ6a+mln:pISt@18Y&q*lpuqmi=AYP<W!49I?BQcI<iN+Mk:y$m%
Q9e+iR N>YFSAejW=4568@GwhhRFh>&E>MAlmjwBbwcd5cyt#BIw@6MCLdN#B$>#Q6AB#SLh9=S<Al^EJ4NlNpS
S<$kyM da!BbT@AiIy#hhLa16^8*8@ylG/Vu:WLF6EnybT+5SLHKTZ6iwu=d4w*l1>Q$9aDyRcp:BGRtja/:=E&
!?QjpS Qmcl:DAd^^*cmW&K==MAA%Z:\ZQ*Qn&LqANel%XTVe&mFWy5k&!cj!*$&A?ph9@+kV1VIB9=656BBG41
pWBMjn <9pP<Wu5ZVSy>\d/EcV6CV5lW%d^NKGwT^pd\n^%4TdHnCd>pVYy:Wpk^!HH<iPYM$5#+W5NtCJ$qDd1
dGIIK? KTTIRWbQQnISq%4^^WPShHa?JbtSRN%^?CcAG?NNmXn8p1SIlJq9d+^J@q+dAW<S=DCGTypp@<*qQZNT
ZNARPP yCJN&M+aNuHFw9+Lc8H1C#Ka9:m41tlN@&9anCi$>L/m*:/<i#a1J&ePT*NupCGEm&%u^aWk+#A5lJ5/
F6BmK@ \Aq8yBy!aklXnD&t51\qnhhRDi>nqc>=\Bq+kQ4Bm>L>nH#4>I6P^=5ck$PjY^J?i\Iy??w4?8ELwh<j
LD%M4Y Q/w!p$=KZylJhXjLGwWZ>uILJlAtMiya1qpelbX4R&:Xc6=u:bD^Mab/\1S=YyNNj/bK#YWlBC!&wY+!
dBjdjy wJ!ii4XetQH4W%$lq&XyYYl6!pKu6d4Rc4W8%K*ZlSc?%H4\*6TG8l8%wji*Cpj#&qD*nN4R/tb61LP%
9NKC\t wXhLR/6WGuyj9XkYl:1dqZAbI>b=PC84FZWjY!8cVpt1FZZ4JbhTJ4=ihcpiSc44m#Yaeq>j$V&IG:@d
XS1ulC ?5#ay6+!ym9KB=>*&=iyNWWVWLm@#8tX?9p?5:DVteLP/neXh41<5LQ/VaYh4mn54H=Vace4Cn84XNlJ
EZt:/1 iAY8yZ!C*6R9YGl8i!PpbJA%#KR8NK>CR@@ZEa#G^LpqtE@%pD6SuAV+nlXB1i:aFje1ta19ZCJPBlM6
/^mnBX h!Y\b!&*Dhn\G&lhW^lYFkeJGQWnD9CKlLYH&JW@=qZj$&<6?/+@6uwci@SY=eE%l@S/1i^!FR5?D>jb
?$TM5F jFttjK&N^By@q<?:%hJq!?G@a^CAAqG!ui<8Dt#Ty<6VDV=tYa41uuHd%>L8YVh<99W>taij9$V:=9YZ
By??LQ %9CJS6F4?T*DQiqhHqQuKbQBEVe+iKCIeERKj5!uJ6mdj:!Fh8*Ip$BH?C!Z^CZ$S8u\NtmESiScHX$&
BJ8ayS &GNupBVLlD\X>qLAKt?DR:iuGMStBq!/CYeIQWYAYLlKyw?e9NncK#nVa*K^E#?5d<RJ#L&Zu^PqZ!@M
m@uIK& V#JLZ*?>8NybXTNXX%PETqwYy$ciK=X=mSJ8AV>JGh5^LEDB%@I#e8P9mZBeyKmbIam&P+^iE/XpFISR
#Rk@N$ Bt9ukL8&Ba/AMGy<8IwHY?>uuaZXbBIT6LIF*^1?emb<XV$@jKXP1QR4ppGFKVEaWhJZp41C#Mk/$J/u
#SqdMi uBG9*wY>imq!WX!=cJy/=c=n5l*XSCpF9*Jahq>aji6?alGBmky^MbSGHyBeHBd9Z+*q*4G^R:RcM<9W
\Nb/Wk LX&pK+b<TVPaiaTT8AKSb<GY65nGG*i#8DCy8I+!HSm158@dhM&LF5CCT&K<6Eu4<VVF\npb8dX:mLhK
8$>j<F +m\iJDG$jk:hw1qwhea/Kn+<VbT:FD<qRhWia>kq>4ppaL+FRHL^5b>c/aib&b?C*LMqS>n>DG6*6at*
G/L<K# MKby9yR$4hn=uSE6PyIjXS/X/nq@@k1l6DKImZKZR@y8*RT/9PQHMk#?LA#yMkY>/4GQinulJXciT\HZ
@=?Ei* /ClJRJCp1#&l+H*?#h!tC5Tw%R\$WX@&FV^jI%t#cqX4jyhQu@4B^1Lu>Z:P<ykbIpuMj!5AaDTiT+hw
RwMcm4 GBhu1L\S%1dlCcy<8$c>6u1Z\T8p#t#ZdEi>R4XZ1<Ym<Q%y&HTRwNjW4aI/^?%wAwQWi^J<ZwG6I4NK
\9wa9% ELXutdYAi1Z4/t1?8%I<&GVZ*!*\jkTcMWcqywDC=tw:=*V9H9mbeuiJmq$G?4S<JWknb%\dP?LqiqN5
THXqj@ Ld+l?e/89w6qbZSi8B$=y:wTdT6L&!LmqG:QWqNZ:MCqjcCw%c\**DThDk!Y$HcBSIDS6w?M1&yq?*dq
b1c91K >N#/%1?dj\\PXpZVDcm@S+ek:TkRFWKD6t!/B+Q*FAqht?qJ%SkBRX%S^>Jl>LmViF@d*GYZSD==Lam^
DW9bLF L85dFtY6pyd:PE+S=CHGcGq^4l8eMVL<m5*:N/6la#JI9@45dqHZtV#9t#cm19&<ZJHWyI9Imm6hbTZY
=9e*Bu t%lw/G$j*BG:Sd46$A*T5qipI:wJVS6da/5de4ewVZn8ZyTY9tumt*YVZ>D>m56F<YTk/=9Q46Q<#e9Q
Zy@9G/ u\P6Z4%Dy8InlBnd&pq%AZi<4cdlNFKFc1DMdVJ8w#*JHaeZF/1<L!=W\ZhG+QW^ZY1*%uS/TbJ4?a5L
uwEmcV P#>9Z4/!NHpLXP/14#PCTIVkNlkQW=Hedb\l<dEJBhWEb=c5>LVM$pVJ1pP^MdWj=QaJlqN/NDQEjLCq
KQNb<A kNJh4i#itF?%t%+kSX#e@pyJ8KtXY:@nEGyKDl\/CTQ!?YaECddQi1>bba&8BF+u9d#!ZY!%\4VLqL=E
QRP/L= 5LYn\w%CFlKThtpERKH&K%PMHAqed8Ym>t?1BPd>Zp4\X!/Zh/tBJ>+SjlcFp@11/n*#5VZJ/nZ#LnXu
dCeRMc pw&tKZFtYKt$Mi:J#b/qCqyN+PP<&@^RqeWEBq+eQ+F*kE1:X1K@MJa$am%9E+#wY\9!#t&1qWQ*SNd4
\pFTh^ Jb?\4:K+k$TGZ4/t*>=khhYPi!$JmnwG/Jj\STM9yiL5lBq^NCHh=D6^4DB4p>%nkKDpD#RRFl8*GPGd
Fd/jBa ^JQt&$V59Ri/++^pX9Vnl?WDSXeyb<IZFL&PqTXFw?T%8AnbWB5Tmk<^=pnZ5VNt%>!aWMy9Vh@pV4wX
k=91!d IQGG6#qKiQiMcFPlk:/*V$cB&Qaa#&AHS?<PYF/ny*WBny&G^Nmy4+tBpT+8DT@4IJLctFK9t>Ye9+SL
jpVS6K <D*6m^jn^w$S>:BF&uWnHZmVj!lKBSdB6^?9B>??ED/*/8j$kCAnQl/9Q49:*&L/VPHD*j>5qQLHleGH
PyH6c9 Mtyc://u4j+YtSJECXD^A^=9dm4$/4PWjBESp:CD<S8*4qS&cWS//ZP\RewBbb4MTHtlK**DuctJy#ma
dwMH\I ^JYF&MLdIHuQ#LG+#&j1a5ekGBunkkpSIHubl:b@$8:<L>=1@+BP$6MWMXjd1>QCqbuZL:k*EbS*8:\l
+?=bN! =/:>IE+uh/1DThCXVh14Z9Sbu9M5$81pAC*9N1*&R!ma!V#y=G+pt!@D1$aZ1bQawJPShXcb88e5A*tB
1HED%! ?==nhK%amq9MjihdAS/\RZe^#a9ZX8QYW+uTQNK9><5TY>H\+jM:cYy5Z6%%9%kN8i94=#n@iq*b>yqT
tdDE#Y LP&8hM*WAGySe//NwX5&m<#&kuC#:TVN%WdXc5QNJeqR8n=nGe5R#ei$JVXJZmua*VpLXlC@T!!QtPwi
m\iD4i X&=u%TT*?/kpN^<b#uR*DQYtXLjV?>GjtG%%hddP*K\hE@AFRb?iwL5!\V\t<q:9\H6J8@%1iXP5C4YZ
:n&6S% Z!dF<i@kFYE8NnZ^jNIM@WViAWZ??iijtpt<@u>8Mdt!&uLNVL^lIFXwIDpyGE\G$pKq9*5mD9SbVeSb
#d*Bkh VRF@Kd<6d:dA9RhCaQ5LwTC!&E$!4F=9pW^w9AwSaX^L#Xq/DQqFcQEGH9H59%jDu9Dw*\*VFG@qY:G+
8\NY<6 <MlIY\ECqJ/$CIkcb:p%%+TAZq!QWYV^51\RL<6quwH6GjNQ=L=F5i:@ZqQ>^*i*wK/9EK@\=w89XkNw
!MGHBP BNb<<?ZL8>ZWn\b<F8mL*D!Jq^k+K4GI=Vcc*/Xt:=MW1/!nPDkCLVitYLVbVn5$eNNB9h!WM+><//qi
MW?h1c ZnKl6eQNIQVL<FXh\!ykbH%=d4p$6nT^ujJJCmVjbt^IF>$en=*t658dV//NL/qjM>/b8T1Ye=9L9GKu
cNY<\p A8\Z4Ziyt+<lwcYci@Y:Pdthq?1@#Akyd<cJ@!G<4KuKhw1%tSy6TciRtQ!JKX!bRh/!>JC/F$YQ=P:@
9<Q69? X!*$$aj?*R%+DIYmZu>tD:&Ik#j$!<8qj91Dq6<pIFaMdh??NWbSFHqu+iYhS9uhDBRa@Ab4MJi*GXY4
5h%lq4 GYV4EyuLn6\%qN1cV&^#%6uFS!/8u#*i:Qan*K91nKW^Ei<kNbI>6S<SWiH+Ctb1E+H8/ZE+unwMRNM:
edD%B% Gcb<\*h5NbJ@wbZAtZeSNckMwNyGtlqj59bp%QMYHRA^a%&M5>q4P$Vq*t#$L#YiT@Rmb*hqAd/+\Cml
5T:NHb ?SQ:WF98MFK#Fe+L!pX9Sq*K&8b#/HbZ:KHubcH?&/I+X<BtE6hmQju/%X4a>^Xqa%=@<X@P@Ap!BTb:
cRLMVn <GSbCTR&R\BbAd4a\!\Vp6*k5@d$\BYn6?AhL1tjqiBe>b#kZEyJpMH1BWiGaRACBR&W*#+P=M6<$E\$
?BY6NJ 4^#?<M?DG@:+\@WpJI9VL%h61JjpDd#?K6@@**ALTZpJCYj9#S$i!+6CwnAwRn$b$j=8QFBMpCB<lZM+
w+=jQY F#&<8Yt*EA*&I1cmC:$+9Bu5jCm8pNDGD?Vba4YpXqC#dIpkawYR&HbFB1PVP\N&Xid4Y8A!LY<8?&+e
:bKemF JNpXmw9KNFd=H6!yZ%KX%%ZaQY+Sy8*t$M5FKbyH<hm:91L?W%/\!E>jF8Q5BS^eK+mBGuZn#GwPiC5n
Fa<hem j:nKQn\+&<8pqtDR+q8yi$:1JTeHN:@KCcZ%Y#dQ%4n1m+L?BR55\iEG?lK9q*Bq5?/WN#X6<iXW<H$!
>!K%n\ jB@LkEZG$/&IZ9%<QuBF9hHH<AFja:G/Ib1kVb?>#lS8>Ral!^SZV:/y5ZGMGLy=JbEi>QZh@S:qA>$k
ji@P%e XphR/pejLde5+j4Iw*YhwGB5yZ8aCNI?tS64aB^b@%MENVH?Xhb%V:F8>T#Yhhha6*wt/:IV+il8$5bj
```

```
-AaVAq ØØ1839591deØØØ2ØØ1b Page 4 of us-constitution.gz

a9XnZ< umZV4lDiQa8Z1%8<:J:D!>WVyL>nLmNXWGl#?CdGn&eTwNTcR^I?FD=ZW8BqaaEMK>ARku*t&=Kp4IWK
umM:L@ dhn>pI5yKQFle\A^NGMk5pSj9n=Q:k^QaT1<jHyN+KMNSYXC1hHqXH#<!yeej<t?F^5waRklh48JX>m^
RFG8^1 S=nAqck5ZcE5Q$41!E/=*!B%yF5?n8u!MctmhpY/*KhK@1IAWGLk/jA?#WGq*Za#@EZY?Xe9<$IS&IZQ
VQ$FiV Q=Mjj1QbVwBn19%+?ZYP4$dD9cM4\Y<p6y>nP:95Gt!V%ac9@JQhw&6\6JWGXXuIke9y*:^=SRFG8ukH
k=qhFK :BMNLNmiJKq%m?J8!%8S$LFeL@M\TSC>XW/9CGdj><y:SCQ$epj18*uu@m<YK&/\yc<AAL--
```