buildtorrent

Claude Heiland-Allen

2007–2019

Contents

1 acinclude.m4

```
##### http://autoconf-archive.cryp.to/ac_c_bigendian_cross.html
#
# SYNOPSIS
#
#   AC_C_BIGENDIAN_CROSS
#
# DESCRIPTION
#
#   Check endianess even when crosscompiling (partially based on the
#   original AC_C_BIGENDIAN).
#
#   The implementation will create a binary, but instead of running the
#   binary it will be grep'ed for some symbols that differ for
#   different endianess of the binary.
#
#   NOTE: The upcoming autoconf 2.53 does include the idea of this
#   macro, what makes it superfluous by then.
#
# LAST MODIFICATION
#
#   2006-10-13
```

```
#
# COPYLEFT
#
#   Copyright (c) 2006 Guido U. Draheim <guidod@gmx.de>
#
#   This program is free software; you can redistribute it and/or
#   modify it under the terms of the GNU General Public License as
#   published by the Free Software Foundation; either version 2 of the
#   License, or (at your option) any later version.
#
#   This program is distributed in the hope that it will be useful, but
#   WITHOUT ANY WARRANTY; without even the implied warranty of
#   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
#   General Public License for more details.
#
#   You should have received a copy of the GNU General Public License
#   along with this program; if not, write to the Free Software
#   Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA
#   02111-1307, USA.
#
#   As a special exception, the respective Autoconf Macro's copyright
#   owner gives unlimited permission to copy, distribute and modify the
#   configure scripts that are the output of Autoconf when processing
#   the Macro. You need not follow the terms of the GNU General Public
#   License when using or distributing such scripts, even though
#   portions of the text of the Macro appear in them. The GNU General
#   Public License (GPL) does govern all other use of the material that
#   constitutes the Autoconf Macro.
#
#   This special exception to the GPL applies to versions of the
#   Autoconf Macro released by the Autoconf Macro Archive. When you
#   make and distribute a modified version of the Autoconf Macro, you
#   may extend this special exception to the GPL to apply to your
#   modified version as well.

AC_DEFUN([AC_C_BIGENDIAN_CROSS],
[AC_CACHE_CHECK(whether byte ordering is bigendian, ac_cv_c_bigendian,
[ac_cv_c_bigendian=unknown
# See if sys/param.h defines the BYTE_ORDER macro.
AC_TRY_COMPILE([#include <sys/types.h>
#include <sys/param.h>], [
#if !BYTE_ORDER || !BIG_ENDIAN || !LITTLE_ENDIAN
 bogus endian macros
#endif], [# It does; now see whether it defined to BIG_ENDIAN or not.
AC_TRY_COMPILE([#include <sys/types.h>
#include <sys/param.h>], [
#if BYTE_ORDER != BIG_ENDIAN
 not big endian
#endif], ac_cv_c_bigendian=yes, ac_cv_c_bigendian=no)])
if test $ac_cv_c_bigendian = unknown; then
AC_TRY_RUN([main () {
  /* Are we little or big endian?  From Harbison&Steele.  */
  union
  {
    long l;
    char c[sizeof (long)];
  } u;
```

```
  u.l = 1;
  exit (u.c[sizeof (long) - 1] == 1);
}], ac_cv_c_bigendian=no, ac_cv_c_bigendian=yes,
[ echo $ac_n "cross-compiling... " 2>&AC_FD_MSG ])
fi])
if test $ac_cv_c_bigendian = unknown; then
AC_MSG_CHECKING(to probe for byte ordering)
[
cat >conftest.c <<EOF
short ascii_mm[] = { 0x4249, 0x4765, 0x6E44, 0x6961, 0x6E53, 0x7953, 0 };
short ascii_ii[] = { 0x694C, 0x5454, 0x656C, 0x6E45, 0x6944, 0x6E61, 0 };
void _ascii() { char* s = (char*) ascii_mm; s = (char*) ascii_ii; }
short ebcdic_ii[] = { 0x89D3, 0xE3E3, 0x8593, 0x95C5, 0x89C4, 0x9581, 0 };
short ebcdic_mm[] = { 0xC2C9, 0xC785, 0x95C4, 0x8981, 0x95E2, 0xA8E2, 0 };
void _ebcdic() { char* s = (char*) ebcdic_mm; s = (char*) ebcdic_ii; }
int main() { _ascii (); _ebcdic (); return 0; }
EOF
] if test -f conftest.c ; then
     if ${CC-cc} -c conftest.c -o conftest.o && test -f conftest.o ; then
        if test `grep -l BIGenDianSyS conftest.o` ; then
           echo $ac_n ' big endian probe OK, ' 1>&AC_FD_MSG
           ac_cv_c_bigendian=yes
        fi
        if test `grep -l LiTTleEnDian conftest.o` ; then
           echo $ac_n ' little endian probe OK, ' 1>&AC_FD_MSG
           if test $ac_cv_c_bigendian = yes ; then
            ac_cv_c_bigendian=unknown;
           else
            ac_cv_c_bigendian=no
           fi
        fi
        echo $ac_n 'guessing bigendian ...  ' >&AC_FD_MSG
     fi
  fi
AC_MSG_RESULT($ac_cv_c_bigendian)
fi
if test $ac_cv_c_bigendian = yes; then
  AC_DEFINE(WORDS_BIGENDIAN, 1, [whether byteorder is bigendian])
  BYTEORDER=4321
else
  BYTEORDER=1234
fi
AC_DEFINE_UNQUOTED(BYTEORDER, $BYTEORDER, [1234 = LIL_ENDIAN, 4321 = BIGENDIAN])
if test $ac_cv_c_bigendian = unknown; then
  AC_MSG_ERROR(unknown endianess - sorry, please pre-set ac_cv_c_bigendian)
fi
])
```

2 AUTHORS

```
Claude Heiland-Allen <claude@mathr.co.uk>
Paul Drain
```

3 buildtorrent.1

```
.TH BUILDTORRENT "1" "April 2010" "0.9~git" "User Commands"
.SH NAME
```

```
buildtorrent \- a torrent file creation program.
.SH SYNOPSIS
.B buildtorrent
[\fIOPTIONS\fR] \fI\-a announceurl input output\fR
.RS 0
.B buildtorrent
[\fIOPTIONS\fR] \fI\-a announceurl \-f filelist \-n name output\fR
.RE
.SH DESCRIPTION
.B buildtorrent
is a torrent file creation program.
Given an announce url and an input file or directory,
.B buildtorrent
generates an output
.I .torrent
file that can be used by torrent clients.
.SH OPTIONS
.TP
.BI \-a\ announce ,\ \-\-announce= url
Announce URL (required).
.TP
.BI \-f\ filelist ,\ \-\-filelist= filelist
A text file (or
.I -
for standard input) containing a list of files to add to the output
torrent file, together with the path to use inside the torrent. One
file is given per line, use
.I /
as path separator for the torrent path, use
.I |
to separate the filesystem path from the torrent path. Backslash
.I \e
can be used to escape newlines and
.I |
characters inside names. For example:
.P
.RS
.nf
.I /data/files/linux.iso|bin/linux.iso
.I /data/files/linux.txt|doc/linux.txt
.fi
.RE
.P
.RS
Using this option requires that the
.I \-\-name
option be used.
.RE
.TP
.BI \-n\ name ,\ \-\-name= name
Specify the name for the torrent. Usage of this option is required when the
.I \-\-filelist
option is used, in which case it specifies the name of the torrent directory.
Usage without a file list overrides the name of the directory or file given
on the command line.
.TP
.BI \-A\ announces ,\ \-\-announcelist= announces
```

```
60  Additional announce URL list.  Use
    .I ,
    to separate outer level lists, and
    .I |
    to separate inner level items; for example:
65  .P
    .RS
    .I a,b1|b2,c
    .RE
    .TP
70  .BI \-w\  webseeds ,\ \-\-webseeds= webseeds
    Additional WebSeed URL list.  Use
    .I ,
    to separate items; for example:
    .P
75  .RS
    .I a,b,c
    .RE
    .TP
    .BI \-l\  length ,\ \-\-piecelength= length
80  Piece length in bytes (default 262144).
    .TP
    .BI \-L\  length ,\ \-\-piecesize= size
    Use 2^size as piece length (default 18) (overrides
    .I \-l
85  ).
    .TP
    .BI \-c\  comment ,\ \-\-comment= comment
    User comment (omitted by default).
    .TP
90  .BI \-p\  private ,\ \-\-private= private
    Private flag (either 0 or 1).
    .TP
    .B \-D, \-\-nodate
    Omit the
95  .I creation date
    field.
    .TP
    .B \-C, \-\-nocreator
    Omit the
100 .I created by
    field.
    .TP
    .B \-m, \-\-md5sum
    Add an
105 .I md5sum
    field for each file.
    .TP
    .BI \-s\  sortorder ,\ \-\-sort= sortorder
    Filename sorting method (only when in filesystem directory
110 mode): any one of Unsorted, Alpha, Version (when available).
    .TP
    .B \-q, \-\-quiet
    Quiet operation with reduced output.
    .TP
115 .B \-v, \-\-verbose
    Verbose operation with increased output (may be used multiple times
```

```
for more verbosity).
.TP
.B \-V, \-\-version
Show the version string.
.TP
.B \-h, \-\-help
Show a help screen with brief usage information.
.SH NOTES
The
.I \-s
and
.I \-S
options have been removed; use
.I \-v
and
.I \-vv
instead.
.SH AUTHORS
Claude Heiland-Allen (claude@mathr.co.uk)
.SH SEE ALSO
.BR createtorrent (1),
.BR mktorrent (1)
.RS 0
http://wiki.theory.org/BitTorrentSpecification
.RS 0
http://bittorrent.org/beps/bep\_0012.html
.RS 0
http://www.getright.com/seedtorrent.html
```

4 buildtorrent.c

```
/*
buildtorrent -- torrent file creation program
Copyright (C) 2007,2008,2009,2010 Claude Heiland-Allen

This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License
as published by the Free Software Foundation; either version 2
of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA.
*/

/* GNU source, for versionsort for scandir */
#define _GNU_SOURCE

#include "config.h"

#include <assert.h>
```

```
#include <ctype.h>
#include <getopt.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/stat.h>
#include <inttypes.h>
#include <dirent.h>
#include <time.h>
/* for correct behaviour ensure this is the POSIX and not the GNU basename:
   "With glibc, one gets the POSIX version of basename() when <libgen.h> is
    included, and the GNU version otherwise." */
#include <libgen.h>

#include "sha1.h"
#include "md5.h"


/*************************************************************************
program version string
*************************************************************************/
#define bt_version "0.9~git"
const char* __version__ = bt_version;

/*************************************************************************
torrent data structure declarations
*************************************************************************/

/* structure for torrent file data */
struct _bt_data;
typedef struct _bt_data *bt_data;

/* error flags */
enum _bt_error {
  BT_ERROR_NONE = 0,
  BT_ERROR_IO,
  BT_ERROR_OVERFLOW,
  BT_ERROR_PARSE
};
typedef enum _bt_error bt_error;

/* attempt to support large files */
typedef int64_t integer;

/* constructors */
bt_data bt_string(const unsigned int length, const char *data);
bt_data bt_integer(const integer number);
bt_data bt_list(void);
bt_data bt_dictionary(void);

/* append to a list */
void bt_list_append(bt_data list, const bt_data data);

/* insert into a dictionary, maintaining sorted keys */
void bt_dictionary_insert(
  bt_data dictionary,
  const unsigned int keylength, const char *keyname,
```

```
  const bt_data data
);

/* deep-copy a structure */
bt_data bt_copy(const bt_data data);

/* write as a torrent file */
bt_error bt_write(FILE *f, const bt_data bd);

/* types of structures */
enum _bt_type {
  BT_TYPE_STRING = 1000,
  BT_TYPE_INTEGER,
  BT_TYPE_LIST,
  BT_TYPE_DICTIONARY
};
typedef enum _bt_type bt_type;

/* string structure */
struct _bt_string {
  unsigned int length;
  char *data;
};

/* integer structure */
struct _bt_integer {
  integer number;
};

/* list structure */
struct _bt_list {
  struct _bt_data *item;
  struct _bt_list *next;
};

/* dictionary structure */
struct _bt_dictionary {
  struct _bt_string *key;
  struct _bt_data *value;
  struct _bt_dictionary *next;
};

/* general structure */
struct _bt_data {
  bt_type type;
  union {
    struct _bt_string *string;
    struct _bt_integer *integer;
    struct _bt_list *list;
    struct _bt_dictionary *dictionary;
  } b;
};

/*************************************************************************
abort on memory allocation failure
*************************************************************************/
void *bt_malloc(size_t size) {
```

```
  void *mem = malloc(size);
  if (!mem) {
    fprintf(stderr, "buildtorrent: out of memory\n");
    exit(1);
  }
  return mem;
}

/**********************************************************************
allocate a new string structure
**********************************************************************/
bt_data bt_string(const unsigned int length, const char *data) {
  bt_data bd = bt_malloc(sizeof(struct _bt_data));
  struct _bt_string *s = bt_malloc(sizeof(struct _bt_string));
  s->data = bt_malloc(length);
  s->length = length;
  memcpy(s->data, data, length);
  bd->type = BT_TYPE_STRING;
  bd->b.string = s;
  return bd;
}
#define bt_string0(s) bt_string(strlen((s)),(s))

/**********************************************************************
allocate a new integer structure
**********************************************************************/
bt_data bt_integer(const integer number) {
  bt_data bd = bt_malloc(sizeof(struct _bt_data));
  struct _bt_integer *n = bt_malloc(sizeof(struct _bt_integer));
  n->number = number;
  bd->type = BT_TYPE_INTEGER;
  bd->b.integer = n;
  return bd;
}

/**********************************************************************
allocate a new empty list structure
invariant: bd->b.list != NULL && last(bd->b.list).{item, next} == NULL
**********************************************************************/
bt_data bt_list(void) {
  bt_data bd = bt_malloc(sizeof(struct _bt_data));
  struct _bt_list *l = bt_malloc(sizeof(struct _bt_list));
  l->item = NULL;
  l->next = NULL;
  bd->type = BT_TYPE_LIST;
  bd->b.list = l;
  return bd;
}

/**********************************************************************
allocate a new empty dictionary structure
invariant: bd->b.dictionary != NULL &&
           last(bd->b.list).{key, value, next} == NULL &&
           ordered ascending by key
**********************************************************************/
bt_data bt_dictionary(void) {
  bt_data bd = bt_malloc(sizeof(struct _bt_data));
```

```
  struct _bt_dictionary *d = bt_malloc(sizeof(struct _bt_dictionary));
  d->key = NULL;
  d->value = NULL;
  d->next = NULL;
  bd->type = BT_TYPE_DICTIONARY;
  bd->b.dictionary = d;
  return bd;
}

/*************************************************************************
precondition: list is a valid list
append an item to a list
the item is not copied
*************************************************************************/
void bt_list_append(bt_data list, const bt_data data) {
  assert(list);
  assert(BT_TYPE_LIST == list->type);
  assert(list->b.list);
  {
  struct _bt_list *current;
  struct _bt_list *end = bt_malloc(sizeof(struct _bt_list));
  end->item = NULL;
  end->next = NULL;
  current = list->b.list;
  while (current->next) {
    current = current->next;
  }
  current->item = data;
  current->next = end;
  }
}

/*************************************************************************
precondition: dictionary is a valid dictionary
insert an item into a dictionary
the value is not copied, the key is copied
maintains an ascending ordering of key names
FIXME: assumes key names are null terminated
*************************************************************************/
void bt_dictionary_insert(
  bt_data dictionary,
  const unsigned int keylength, const char *keyname,
  const bt_data data
) {
  assert(dictionary);
  assert(BT_TYPE_DICTIONARY == dictionary->type);
  assert(dictionary->b.dictionary);
  {
  struct _bt_dictionary *current;
  struct _bt_dictionary *lastcurrent = NULL;
  struct _bt_dictionary *insertee;
  struct _bt_string *key = bt_malloc(sizeof(struct _bt_string));
  key->data = bt_malloc(keylength);
  insertee = bt_malloc(sizeof(struct _bt_dictionary));
  memcpy(key->data, keyname, keylength);
  key->length = keylength;
  insertee->key = key;
```

```
255     insertee->value = data;
        insertee->next = NULL;
        current = dictionary->b.dictionary;
        while (
          current->next && current->key && (0 < strcmp(keyname, current->key->data))
260     ) {
          lastcurrent = current;
          current = current->next;
        }
        if (lastcurrent) {
265       insertee->next = current;
          lastcurrent->next = insertee;
        } else {
          insertee->next = dictionary->b.dictionary;
          dictionary->b.dictionary = insertee;
270     }
        }
      }

      #define bt_dictionary_insert0(dict,key,data) \
275   bt_dictionary_insert((dict),strlen((key)),(key),(data))

      /*************************************************************************
      precondition: data is valid
      deep copy a data structure
280   FIXME: doesn't handle dictionaries yet
      *************************************************************************/
      bt_data bt_copy(const bt_data data) {
        assert(data);
        switch (data->type) {
285     case (BT_TYPE_STRING): {
          return bt_string(data->b.string->length, data->b.string->data);
        }
        case (BT_TYPE_INTEGER): {
          return bt_integer(data->b.integer->number);
290     }
        case (BT_TYPE_LIST): {
          struct _bt_list *current;
          bt_data list = bt_list();
          current = data->b.list;
295       while (current && current->item) {
            bt_list_append(list, bt_copy(current->item));
            current = current->next;
          }
          return list;
300     }
        default: {
          fprintf(stderr, "buildtorrent: BUG! can't copy this\n");
          exit(1);
        }
305     }
      }

      /*************************************************************************
      precondition: s is valid
310   write a string in torrent encoding
      *************************************************************************/
```

```
bt_error bt_write_string(FILE *f, const struct _bt_string *s) {
  assert(s);
  assert(s->data);
  if (0 > fprintf(f, "%d:", s->length)) {
    return BT_ERROR_IO;
  }
  if (1 != fwrite(s->data, s->length, 1, f)) {
    return BT_ERROR_IO;
  }
  return BT_ERROR_NONE;
}

/*****************************************************************************
precondition: bd is valid
write a data structure in torrent encoding
*****************************************************************************/
bt_error bt_write(FILE *f, const bt_data bd) {
  assert(bd);
  switch (bd->type) {
  case (BT_TYPE_STRING): {
    return bt_write_string(f, bd->b.string);
    break;
  }
  case (BT_TYPE_INTEGER): {
    assert(bd->b.integer);
    if (0 > fprintf(f, "i%" PRId64 "e", bd->b.integer->number)) {
      return BT_ERROR_IO;
    }
    break;
  }
  case (BT_TYPE_LIST): {
    struct _bt_list *current;
    bt_error err;
    current = bd->b.list;
    if (0 > fprintf(f, "l")) {
      return BT_ERROR_IO;
    }
    while (current->next) {
      if (current->item) {
        if ((err = bt_write(f, current->item))) {
          return err;
        }
      }
      current = current->next;
    }
    if (0 > fprintf(f, "e")) {
      return BT_ERROR_IO;
    }
    break;
  }
  case (BT_TYPE_DICTIONARY): {
    struct _bt_dictionary *current;
    bt_error err;
    current = bd->b.dictionary;
    if (0 > fprintf(f, "d")) {
      return BT_ERROR_IO;
    }
```

```
    while (current->next) {
      if (current->key) {
        if ((err = bt_write_string(f, current->key))) {
          return err;
        }
        if ((err = bt_write(f, current->value))) {
          return err;
        }
      }
      current = current->next;
    }
    if (0 > fprintf(f, "e")) {
      return BT_ERROR_IO;
    }
    break;
  }
  default: {
    fprintf(stderr, "buildtorrent: BUG! can't write this\n");
    exit(1);
    break;
  }
  }
  return BT_ERROR_NONE;
}

/*************************************************************************
pretty print torrent data
*************************************************************************/
void bt_show(
  bt_data bd, int piecestoo, int indent, int indentstep, int comma
) {
  int i;
  if (!bd) {
    for(i = 0; i < indent; i++) {
      printf(" ");
    }
    printf("NULL%s\n", comma ? "," : "");
    return;
  }
  switch (bd->type) {
  case (BT_TYPE_STRING): {
    char strbuf[512];
    int len = bd->b.string->length > 500 ? 500 : bd->b.string->length;
    memcpy(strbuf, bd->b.string->data, len);
    strbuf[len] = '\0';
    for(i = 0; i < indent; i++) {
      printf(" ");
    }
    printf(
      "\"%s\"%s%s\n", strbuf, comma ? "," : "",
      len == 500 ? " // truncated!" : ""
    );
    break;
  }
  case (BT_TYPE_INTEGER): {
    for(i = 0; i < indent; i++) {
      printf(" ");
```

```
    }
    printf("%" PRId64 "%s\n", bd->b.integer->number, comma ? "," : "");
    break;
  }
  case (BT_TYPE_LIST): {
    struct _bt_list *current;
    for(i = 0; i < indent; i++) {
      printf(" ");
    }
    printf("[\n");
    current = bd->b.list;
    while (current->next) {
      bt_show(
        current->item, piecestoo,
        indent + indentstep, indentstep,
        current->next->next != NULL
      );
      current = current->next;
    }
    for(i = 0; i < indent; i++) {
      printf(" ");
    }
    printf("]%s\n", comma ? "," : "");
    break;
  }
  case (BT_TYPE_DICTIONARY): {
    struct _bt_dictionary *current;
    char strbuf[512];
    int len;
    for(i = 0; i < indent; i++) {
      printf(" ");
    }
    printf("{\n");
    current = bd->b.dictionary;
    while (current->next) {
      len = current->key->length > 500 ? 500 : current->key->length;
      memcpy(strbuf, current->key->data, len);
      strbuf[len] = '\0';
      for(i = 0; i < indent + indentstep; i++) {
        printf(" ");
      }
      printf("\"%s\" =>%s\n", strbuf, len == 500 ? " // truncated!" : "");
      if (strcmp("pieces", strbuf) == 0) {
        if (piecestoo) {
          char *hexdigits = "0123456789abcdef";
          printf("---------------------------------------");
          for (i = 0; i < current->value->b.string->length; i++) {
            if ((i % 20) == 0) {
              printf("\n");
            }
            printf("%c%c",
              hexdigits[(current->value->b.string->data[i] & 0xF0) >> 4],
              hexdigits[(current->value->b.string->data[i] & 0x0F)]
            );
          }
          printf("\n---------------------------------------\n");
        } else {
```

```
            for(i = 0; i < indent + indentstep + indentstep; i++) {
              printf(" ");
485           }
              printf(
                "\"...\"%s // pieces not shown\n", current->next->next ? "," : ""
              );
            }
490       } else {
            bt_show(
              current->value, piecestoo, indent + indentstep + indentstep,
              indentstep, current->next->next != NULL
            );
495       }
          current = current->next;
        }
        for(i = 0; i < indent; i++) {
          printf(" ");
500     }
        printf("}%s\n", comma ? "," : "");
        break;
      }
      }
505 }


    /*************************************************************************
    file list data structure
510 the first in the list is a dummy
    *************************************************************************/
    struct _bt_file_list {
      char *file;
      bt_data path;
515   struct _bt_file_list *next;
    };
    typedef struct _bt_file_list *bt_file_list;

    /*************************************************************************
520 create the dummy first list element
    *************************************************************************/
    bt_file_list bt_create_file_list() {
      bt_file_list flist = bt_malloc(sizeof(struct _bt_file_list));
      flist->file = NULL;
525   flist->path = NULL;
      flist->next = NULL;
      return flist;
    }

530 /*************************************************************************
    precondition: flist and file must be valid
    prepend to the file list
    copies the arguments
    path may be NULL, but only in single file mode
535 *************************************************************************/
    void bt_file_list_prepend(
      bt_file_list flist, const char *file, bt_data path
    ) {
      assert(flist);
```

```
540     assert(file);
        {
        bt_file_list node = bt_malloc(sizeof(struct _bt_file_list));
        node->file = bt_malloc(strlen(file) + 1);
        memcpy(node->file, file, strlen(file) + 1);
545     if (path) {
          node->path = bt_copy(path);
        } else {
          node->path = NULL;
        }
550     node->next = flist->next;
        flist->next = node;
        }
      }

555
      /*************************************************************************
      annotated file list data structure
      the first in the list is a dummy
      *************************************************************************/
560   struct _bt_afile_list {
        char *file;
        bt_data path;
        integer length;
        bt_data md5sum;
565     struct _bt_afile_list *next;
      };
      typedef struct _bt_afile_list *bt_afile_list;

      /*************************************************************************
570   create the dummy first list element
      *************************************************************************/
      bt_afile_list bt_create_afile_list() {
        bt_afile_list aflist = bt_malloc(sizeof(struct _bt_afile_list));
        aflist->file = NULL;
575     aflist->path = NULL;
        aflist->length = 0;
        aflist->md5sum = NULL;
        aflist->next = NULL;
        return aflist;
580   }

      /*************************************************************************
      preconditions: aflist and fnode are valid
      prepend to the annotated file list
585   aflist is an annotated file list to prepend to
      fnode is a file list node to annotate
      length is the length to annotate with
      md5sum is the md5sum to annotate with (may be NULL)
      NOTE does NOT create a new copy of the existing data
590   *************************************************************************/
      void bt_afile_list_prepend(
        bt_afile_list aflist, bt_file_list fnode, integer length, bt_data md5sum
      ) {
        assert(aflist);
595     assert(fnode);
        {
```

```
    bt_afile_list node = bt_malloc(sizeof(struct _bt_afile_list));
    node->file = fnode->file;
    node->path = fnode->path;
    node->length = length;
    node->md5sum = md5sum;
    node->next = aflist->next;
    aflist->next = node;
    }
}


/*****************************************************************************
append a file to a path string
path must be at least length maxlength bytes
returns an error flag
*****************************************************************************/
bt_error bt_joinpath(size_t maxlength, char *path, const char *file) {
  if (strlen(path) + strlen(file) + 1 + 1 > maxlength) {
    return BT_ERROR_OVERFLOW;
  } else {
    size_t oldlength = strlen(path);
    if (oldlength > 0) {
      path[oldlength] = '/';
      memcpy(path + oldlength + 1, file, strlen(file) + 1);
    } else {
      memcpy(path, file, strlen(file) + 1);
    }
    return BT_ERROR_NONE;
  }
}


/*****************************************************************************
md5sum a file
*****************************************************************************/
bt_data bt_md5sum_file(const char *filename, integer length) {
  struct MD5Context context;
  char *hexdigits = "0123456789abcdef";
  unsigned char *buffer = NULL;
  unsigned char digest[16];
  char hexdump[33];
  integer buflen = 262144;
  integer size = 0;
  integer left = length;
  integer i;
  FILE *file = NULL;
  if (!filename) {
    return NULL;
  }
  buffer = bt_malloc(buflen);
  if (!(file = fopen(filename, "rb"))) {
    free(buffer);
    return NULL;
  }
  MD5Init(&context);
  while (left > 0) {
    if (left > buflen) {
```

```
        size = buflen;
655     } else {
        size = left;
      }
      if (1 != fread(buffer, size, 1, file)) {
        free(buffer);
660     fclose(file);
        return NULL;
      }
      MD5Update(&context, buffer, size);
      left -= size;
665   }
    MD5Final(digest, &context);
    for (i = 0; i < 16; ++i) {
      hexdump[i+i+0] = hexdigits[(digest[i] & 0xF0) >> 4];
      hexdump[i+i+1] = hexdigits[(digest[i] & 0x0F)];
670   }
    free(buffer);
    fclose(file);
    return bt_string(32, hexdump);
  }
675

  /*****************************************************************************
  find files recursively
  directory is the initial directory
680 path is a buffer of maxlength bytes to store the accumulated path
  pathlist is the accumulated path exploded as a list
  files is a file list to add the found files to
  returns an error flag
  *****************************************************************************/
685 bt_error bt_find_files(
    int (*compar)(const struct dirent **, const struct dirent **),
    size_t maxlength, char *path, bt_data pathlist, bt_file_list files
  ) {
    struct dirent **entry;
690 bt_data filename = NULL;
    bt_data filepath = NULL;
    bt_error err = BT_ERROR_NONE;
    int i, n;
    /* use scandir instead of readdir for sorted listings */
695 i = scandir(path, &entry, 0, compar);
    if (i >= 0) {
    for (n = 0; n < i; n++) {
      if (strcmp(entry[n]->d_name, ".") && strcmp(entry[n]->d_name, "..")) {
        struct stat s;
700     size_t oldlength = strlen(path);
        if (!(err = bt_joinpath(maxlength, path, entry[n]->d_name))) {
          if (!stat(path, &s)) {
            if (S_ISREG(s.st_mode)) {
              filename = bt_string0(entry[n]->d_name);
705           filepath = bt_copy(pathlist);
              bt_list_append(filepath, filename);
              bt_file_list_prepend(files, path, filepath);
            } else if (S_ISDIR(s.st_mode)) {
              filename = bt_string0(entry[n]->d_name);
710           filepath = bt_copy(pathlist);
```

```
            bt_list_append(filepath, filename);
            err = bt_find_files(compar, maxlength, path, filepath, files);
          } else {
            /* FIXME neither regular file nor directory, what to do? */
715       }
        } else {
          err = BT_ERROR_IO;
        }
      }
720   path[oldlength] = '\0';
    }
    if (err) {
      break;
    }
725 }
  } else {
    err = BT_ERROR_IO;
  }
  return err;
730 }


/*****************************************************************************
annotate a file list
735 files is the file list to annotate
afiles is the file list to add to
returns an error flag
*****************************************************************************/
bt_error bt_annotate_files(
740   bt_file_list files, bt_afile_list afiles
) {
  bt_error err = BT_ERROR_NONE;
  bt_file_list fnode = files;
  while ((fnode = fnode->next)) {
745   struct stat s;
    if (!stat(fnode->file, &s)) {
      if (S_ISREG(s.st_mode)) {
        integer length = s.st_size;
        bt_afile_list_prepend(afiles, fnode, length, NULL);
750   } else {
        err = BT_ERROR_IO;
      }
    } else {
      err = BT_ERROR_IO;
755 }
    if (err) {
      break;
    }
  }
760 return err;
}


/*****************************************************************************
765 convert an annotated file list into torrent format
aflist is the list to convert
returns the torrent format file list
```

```
**************************************************************************/
bt_data bt_afile_list_info(bt_afile_list aflist) {
  bt_afile_list node;
  bt_data files;
  files = bt_list();
  node = aflist;
  while ((node = node->next)) {
    bt_data file;
    bt_data filesize;
    bt_data filepath;
    bt_data md5sum;
    file = bt_dictionary();
    filesize = bt_integer(node->length);
    if (!(filepath = node->path)) {
      fprintf(stderr, "buildtorrent: BUG! path missing from node\n");
      exit(1);
    }
    bt_dictionary_insert0(file, "length", filesize);
    bt_dictionary_insert0(file, "path", filepath);
    if ((md5sum = node->md5sum)) {
      bt_dictionary_insert0(file, "md5sum", md5sum);
    }
    bt_list_append(files, file);
  }
  return files;
}

/**************************************************************************
draw progressbar
**************************************************************************/
void bt_progressbar(integer piece, integer count) {
  integer oldblocks = ((piece - 1) * 50) / count;
  integer blocks = (piece * 50) / count;
  integer i;
  char s[53];
  if (blocks != oldblocks) {
    s[0] = '[';
    for (i = 0; i < 50; i++) {
      s[i+1] = (i < blocks) ? '=' : ((i == blocks) ? '>' : ' ');
    }
    s[51] = ']';
    s[52] = '\0';
    fprintf(
      stderr,
      "\b\b\b\b\b\b\b\b\b\b"
      "\b\b\b\b\b\b\b\b\b\b"
      "\b\b\b\b\b\b\b\b\b\b"
      "\b\b\b\b\b\b\b\b\b\b"
      "\b\b\b\b\b\b\b\b\b\b"
      "\b\b%s",
      s
    );
  }
}
```

```
825 /**************************************************************************
    hash files
    return piece hash as a string
    data side effect: if domd5sum then add md5sums to aflist
    output side effect: if verbose then show file summary
830 **************************************************************************/
    bt_data bt_hash_pieces(
      bt_afile_list aflist, integer size, int domd5sum, int verbose
    ) {
      bt_afile_list node = NULL;
835   char *hashdata = NULL;
      integer total = 0;
      unsigned char *buffer = NULL;
      unsigned char *bufptr = NULL;
      integer remain = size;
840   integer left;
      FILE *file = NULL;
      integer piececount;
      integer i;
      sha1_byte digest[SHA1_DIGEST_LENGTH];
845   if (!aflist) {
        return NULL;
      }
      buffer = bt_malloc(size);
      node = aflist;
850   while ((node = node->next)) {
        if (verbose) {
          fprintf(stderr, "%20" PRId64 " : %s\n", node->length, node->file);
        }
        if (domd5sum) {
855       if (!(node->md5sum = bt_md5sum_file(node->file, node->length))) {
            fprintf(
              stderr,
              "buildtorrent: error computing md5sum for \"%s\"\n",
              node->file
860         );
          }
        }
        total += node->length;
      }
865   piececount = (total + size - 1) / size; /* ceil(total/size) */
      if (piececount <= 0) { /* FIXME: no idea what to do if there's no data */
        free(buffer);
        fprintf(stderr, "torrent has no data, aborting!\n");
        return NULL;
870   }
      if (verbose) {
        fprintf(stderr, "hashing %" PRId64 " pieces\n", piececount);
        fprintf(stderr, "["
                        "          "
875                     "          "
                        "          "
                        "          "
                        "          "
                        "]");
880   }
      hashdata = bt_malloc(piececount * SHA1_DIGEST_LENGTH);
```

```
    node = aflist->next;
    file = fopen(node->file, "rb");
    if (!file) {
885   free(buffer);
      return NULL;
    }
    left = node->length;
    bufptr = buffer;
890 for (i = 0; i < piececount; ++i) {
      do {
        if (left <= remain) {
          /* take all */
          if (left != 0) { /* don't fail on empty files */
895         if (1 != fread(bufptr, left, 1, file)) {
              fclose(file);
              free(buffer);
              return NULL;
            }
900         bufptr += left;
            remain -= left;
            fclose(file);
            file = NULL;
          }
905       node = node->next;
          if (node) {
            file = fopen(node->file, "rb");
            if (!file) {
              free(buffer);
910           return NULL;
            }
            left = node->length;
          }
        } else { /* left > remain */
915       /* take as much as we can */
          if (remain != 0) { /* don't fail on empty files */
            if (1 != fread(bufptr, remain, 1, file)) {
              free(buffer);
              return NULL;
920         }
            bufptr += remain;
            left -= remain;
            remain = 0;
          }
925     }
      } while (remain != 0 && node);
      if (!node && i != piececount - 1) {
        /* somehow the pieces don't add up */
        if (file) {
930       fclose(file);
        }
        free(buffer);
        return NULL;
      }
935   /* remain == 0 || i == piececount - 1 */
      if (verbose) {
        bt_progressbar(i, piececount);
      }
```

```
      SHA1(buffer, size - remain, digest);
      memcpy(hashdata + i * SHA1_DIGEST_LENGTH, digest, SHA1_DIGEST_LENGTH);
      bufptr = buffer;
      remain = size;
    }
    if (verbose) {
      bt_progressbar(piececount, piececount);
      fprintf(stderr, "\n");
    }
    return bt_string(SHA1_DIGEST_LENGTH * piececount, hashdata);
  }

  /*************************************************************************
  parse an announce list
  format = "url|url|url,url|url|url,url|url|url"
  *************************************************************************/
  bt_data bt_parse_announcelist(const char *urls) {
    bt_data announcelist;
    bt_data tier;
    const char *s;
    const char *t;
    const char *t1;
    const char *t2;
    if (!urls) {
      return NULL;
    }
    if (strcmp("", urls) == 0) {
      return NULL;
    }
    announcelist = bt_list();
    s = urls;
    tier = bt_list();
    do {
      t = NULL;
      t1 = strchr(s, '|');
      t2 = strchr(s, ',');
      if (!t1 && !t2) {
        t = s + strlen(s);
      } else if (!t1) {
        t = t2;
      } else if (!t2) {
        t = t1;
      } else {
        t = (t1 < t2) ? t1 : t2;
      }
      if (t <= s) {
        return NULL;
      }
      bt_list_append(tier, bt_string(t - s, s));
      if (t[0] == ',' || t[0] == '\0') {
        bt_list_append(announcelist, tier);
        if (t[0] != '\0') {
          tier = bt_list();
        } else {
          tier = NULL;
        }
```

```
    }
    s = t + 1;
  } while (t[0] != '\0');
  return announcelist;
}

/*************************************************************************
parse a webseed list
format = "url,url,url"
*************************************************************************/
bt_data bt_parse_webseedlist(const char *urls) {
  bt_data webseedlist;
  const char *s;
  const char *t;
  const char *t2;
  if (!urls) {
    return NULL;
  }
  if (strcmp("", urls) == 0) {
    return NULL;
  }
  webseedlist = bt_list();
  s = urls;
  do {
    t = NULL;
    t2 = strchr(s, ',');
    if (!t2) {
      t = s + strlen(s);
    } else {
      t = t2;
    }
    if (t <= s) {
      return NULL;
    }
    bt_list_append(webseedlist, bt_string(t - s, s));
    s = t + 1;
  } while (t[0] != '\0');
  return webseedlist;
}

/*************************************************************************
read and parse a filelist file
line format = "real/file/system/path|tor/rent/path/file"
delimiters: '|', '\n'
escape character: '\\'
*************************************************************************/
enum _bt_parse_state {
  BT_PARSE_LEFT = 0,
  BT_PARSE_LEFT_ESCAPED,
  BT_PARSE_RIGHT,
  BT_PARSE_RIGHT_ESCAPED,
  BT_PARSE_OK,
  BT_PARSE_ERROR
};
typedef enum _bt_parse_state bt_parse_state;
bt_error bt_parse_filelist(bt_file_list flist, FILE *infile) {
  char filename[8192];
```

```
  char torrname[8192];
  bt_data torrpath = NULL;
1055  int c;
  int i = 0;
  int state = BT_PARSE_LEFT;
  while(state != BT_PARSE_OK && state != BT_PARSE_ERROR) {
    c = getc(infile);
1060    switch (state) {
    case (BT_PARSE_LEFT):
      if (c < 0) {
        state = BT_PARSE_OK;
      } else if (c == '\\') {
1065      state = BT_PARSE_LEFT_ESCAPED;
      } else if (c == '|') {
        filename[i] = '\0';
        i = 0;
        torrpath = bt_list();
1070      state = BT_PARSE_RIGHT;
      } else {
        filename[i++] = c;
        if (i > 8190) {
          state = BT_PARSE_ERROR;
1075      } else {
          state = BT_PARSE_LEFT;
        }
      }
      break;
1080  case (BT_PARSE_LEFT_ESCAPED):
      if (c < 0) {
        state = BT_PARSE_ERROR;
      } else {
        filename[i++] = c;
1085    if (i > 8190) {
          state = BT_PARSE_ERROR;
        } else {
          state = BT_PARSE_LEFT;
        }
1090  }
      break;
    case (BT_PARSE_RIGHT):
      if (c < 0) {
        state = BT_PARSE_ERROR;
1095  } else if (c == '\\') {
        state = BT_PARSE_RIGHT_ESCAPED;
      } else if (c == '/' || c == '\n') {
        bt_data torrnamestr;
        torrname[i] = '\0';
1100    i = 0;
        if (0 == strcmp("", torrname)) {
          state = BT_PARSE_ERROR;
        } else {
          torrnamestr = bt_string0(torrname);
1105      bt_list_append(torrpath, torrnamestr);
          if (c == '\n') {
            bt_file_list_prepend(flist, filename, torrpath);
            state = BT_PARSE_LEFT;
          } else {
```

```
1110              state = BT_PARSE_RIGHT;
                }
              }
            } else {
              torrname[i++] = c;
1115          if (i > 8190) {
                state = BT_PARSE_ERROR;
              } else {
                state = BT_PARSE_RIGHT;
              }
1120        }
            break;
          case (BT_PARSE_RIGHT_ESCAPED):
            if (c < 0) {
              state = BT_PARSE_ERROR;
1125        } else {
              torrname[i++] = c;
              if (i > 8190) {
                state = BT_PARSE_ERROR;
              } else {
1130            state = BT_PARSE_RIGHT;
              }
            }
            break;
          default:
1135        fprintf(
              stderr,
              "buildtorrent: BUG! internal error parsing file list (%d)\n",
              state
            );
1140        exit(1);
          }
        }
        if (state == BT_PARSE_OK) {
          return BT_ERROR_NONE;
1145    } else {
          return BT_ERROR_PARSE;
        }
      }

1150  /*************************************************************************
      show usage message
      *************************************************************************/
      void bt_usage(void) {
        printf(
1155      "Usage:\n"
          "  buildtorrent [OPTIONS] -a announceurl input output\n"
          "  buildtorrent [OPTIONS] -a announceurl -f filelist -n name output\n"
          "\n"
          "options:\n"
1160      "--announce      -a  <announce>   : announce url (required)\n"
          "--filelist      -f  <filelist>   : external file list (requires '-n')\n"
          "--name          -n  <name>       : torrent name, default based on input\n"
          "--announcelist  -A  <announces>  : announce url list (format: a,b1|b2,c)\n"
          "--webseeds      -w  <webseeds>   : webseed url list (format: a,b,c)\n"
1165      "--piecelength   -l  <length>     : piece length in bytes, default 262144\n"
          "--piecesize     -L  <size>       : use 2^size as piece length, default 18\n↙
```

```
        ↳ "
    "--comment       -c <comment>    : user comment, omitted by default\n"
    "--private       -p <private>    : private flag, either 0 or 1\n"
    "--nodate        -D              : omit 'creation date' field\n"
    "--nocreator     -C              : omit 'created by' field\n"
    "--md5sum        -m              : add an 'md5sum' field for each file\n"
    "--sort          -s <sortorder>  : sort files ([U]nsorted, "
#ifdef _GNU_SOURCE
    "[A]lpha, [V]ersion)\n"
#else
    "[A]lpha)\n"
#endif
    "--quiet         -q              : quiet operation\n"
    "--verbose       -v              : verbose. More -v means more verbose\n"
    "--version       -V              : show version of buildtorrent\n"
    "--help          -h              : show this help screen\n"
  );
}

/*****************************************************************************
main program
*****************************************************************************/
int main(int argc, char **argv) {

  char *url = NULL;
  char *urls = NULL;
  char *wurls = NULL;
  char *inname = NULL;
  char *nameflag = NULL;
  char *namebase = NULL;
  char *outfile = NULL;
  char *commentstr = NULL;
  char *filelistfilename = NULL;
#ifdef _GNU_SOURCE
  char sort = 'v'; /* use version sort, if available */
#else
  char sort = 'a'; /* otherwise use alpha sort */
#endif
  /* default to no sorting (as before) */
  int (*compar)(const struct dirent **, const struct dirent **) = 0;
  int lplen = -1;
  unsigned int plen = 262144;
  int verbose = 1;
  int nodate = 0;
  int nocreator = 0;
  int privated = 0;
  int privateopt = 0;
  int domd5sum = 0;
  int show = 0;
  int slen;
  int i;

  FILE *output = NULL;
  bt_data torrent = NULL;
  bt_data announcelist = NULL;
  bt_data webseedlist = NULL;
  bt_data info = NULL;
```

```
  bt_data pieces = NULL;
  bt_data pathlist = NULL;

  int multifile = 0;
  bt_file_list flist = NULL;
  bt_afile_list aflist = NULL;

  struct stat s;
  char path[8192];
  char nametemp[8192];

  while (1) {
    int optidx = 0;
    static struct option options[] = {
      { "announce", 1, 0, 'a' },
      { "filelist", 1, 0, 'f' },
      { "name", 1, 0, 'n' },
      { "announcelist", 1, 0, 'A' },
      { "webseeds", 1, 0, 'w' },
      { "piecelength", 1, 0, 'l' },
      { "piecesize", 1, 0, 'L' },
      { "comment", 1, 0, 'c' },
      { "private", 1, 0, 'p' },
      { "sort", 1, 0, 's' },
      { "nodate", 0, 0, 'D' },
      { "nocreator", 0, 0, 'C' },
      { "md5sum", 0, 0, 'm' },
      { "quiet", 0, 0, 'q' },
      { "verbose", 0, 0, 'v' },
      { "version", 0, 0, 'V' },
      { "help", 0, 0, 'h' },
      { 0, 0, 0, 0 }
    };
    char c = getopt_long(argc, argv, "hVqvmCDs:a:f:n:A:w:l:L:c:p:", options, &↙
        ↳ optidx );
    if (c == -1) {
      break;
    }
    switch (c) {
    case ('?'):
      return 1;
    case ('a'):
      url = optarg;
      break;
    case ('f'):
      filelistfilename = optarg;
      break;
    case ('n'):
      nameflag = optarg;
      break;
    case ('A'):
      urls = optarg;
      break;
    case ('w'):
      wurls = optarg;
      break;
    case ('l'):
```

```
        plen = atoi(optarg);
1280    break;
      case ('L'):
        lplen = atoi(optarg);
        break;
      case ('c'):
1285    commentstr = optarg;
        break;
      case ('p'):
        privated = 1;
        privateopt = (strcmp(optarg, "0") == 0) ? 0 : 1;
1290    break;
      case ('s'):
        sort = tolower(optarg[0]);
        switch(sort) {
        default:
1295  #ifdef _GNU_SOURCE
        case 'v':
          /* read directory entry in "version" order, ie. 1 2 10, not 1 10 2 */
          compar = versionsort;
          break;
1300  #endif
        case 'a':
        case 's':
          compar = alphasort;
          break;
1305    case 'u':
          /* read directory entry normal order (directory order, seemingly random)↙
              ↳ */
          compar = 0;
          break;
        }
1310    break;
      case ('D'):
        nodate = 1;
        break;
      case ('C'):
1315    nocreator = 1;
        break;
      case ('m'):
        domd5sum = 1;
        break;
1320  case ('v'):
        show++;
        if(show > 2)
          show=2;
        break;
1325  case ('q'):
        verbose = 0;
        break;
      case ('V'):
        printf(
1330      "buildtorrent " bt_version "\n"
          "Copyright (C) 2007-2010 Claude Heiland-Allen <claude@mathr.co.uk>\n"
          "License GPLv2+: GNU GPL version 2 or later <http://gnu.org/licenses/gpl↙
              ↳ .html>\n"
        );
```

```
      return 0;
    case ('h'):
      bt_usage();
      return 0;
    }
  }
  if (!url) {
    fprintf(stderr, "buildtorrent: announce url required\n");
    fprintf(stderr, "Try `buildtorrent --help' for more information.\n");
    return 1;
  }
  if (0 <= lplen && lplen < 31) {
    plen = 1 << lplen;
  }
  if (plen <= 0) { /* avoid division by zero */
    fprintf(stderr, "buildtorrent: piece length must be greater than 0\n");
    return 1;
  }

  if (filelistfilename) {
    if (optind + 1 < argc) {
      fprintf(stderr, "buildtorrent: too many arguments\n");
      fprintf(stderr, "Try `buildtorrent --help' for more information.\n");
      return 1;
    }
    if (optind + 1 > argc) {
      fprintf(stderr, "buildtorrent: too few arguments\n");
      fprintf(stderr, "Try `buildtorrent --help' for more information.\n");
      return 1;
    }
    if (!nameflag) {
      fprintf(stderr, "buildtorrent: missing '-n', required when using '-f'\n");
      fprintf(stderr, "Try `buildtorrent --help' for more information.\n");
      return 1;
    }
    inname = NULL;
    outfile = argv[optind];
  } else {
    if (optind + 2 < argc) {
      fprintf(stderr, "buildtorrent: too many arguments\n");
      fprintf(stderr, "Try `buildtorrent --help' for more information.\n");
      return 1;
    }
    if (optind + 2 > argc) {
      fprintf(stderr, "buildtorrent: too few arguments\n");
      fprintf(stderr, "Try `buildtorrent --help' for more information.\n");
      return 1;
    }
    inname  = argv[optind];
    outfile = argv[optind + 1];
  }

  /* handle paths correctly (note: requires POSIX basename(), not GNU) */
  if (inname) {
    if (strlen(inname) > 8190) {
      fprintf(stderr, "buildtorrent: 'input' argument too long\n");
      return 1;
```

```
    }
    strncpy(nametemp, inname, 8191);
    nametemp[8191] = '\0';
    namebase = basename(nametemp);
    slen = strlen(namebase);
    for (i = 0; i < slen; ++i) {
      if (namebase[i] == '/') {
        fprintf(
          stderr,
          "buildtorrent: BUG! input (\"%s\") munged (\"%s\") contains '/'.\n",
          inname,
          namebase
        );
        return 1;
      }
    }
  }

  if (inname) {
    if (stat(inname, &s)) {
      fprintf(stderr, "buildtorrent: could not stat \"%s\"\n", inname);
      return 1;
    }
  }

  torrent = bt_dictionary();
  info = bt_dictionary();
  bt_dictionary_insert0(
    info, "name", bt_string0(nameflag ? nameflag : namebase)
  );
  bt_dictionary_insert0(info, "piece length", bt_integer(plen));
  if (urls) {
    if (!(announcelist = bt_parse_announcelist(urls))) {
      fprintf(stderr, "buildtorrent: error parsing announce-list argument\n");
      return 1;
    }
  }
  if (wurls) {
    if (!(webseedlist = bt_parse_webseedlist(wurls))) {
      fprintf(stderr, "buildtorrent: error parsing webseed list argument\n");
      return 1;
    }
  }
  flist = bt_create_file_list();
  aflist = bt_create_afile_list();

  if (inname && S_ISDIR(s.st_mode)) {

    multifile = 1;
    pathlist = bt_list();
    memcpy(path, inname, strlen(inname) + 1);
    if (bt_find_files(compar, 8192, path, pathlist, flist)) {
      fprintf(stderr, "buildtorrent: error finding files\n");
      return 1;
    }

  } else if (inname && S_ISREG(s.st_mode)) {
```

```
    multifile = 0;
1450    bt_file_list_prepend(flist, inname, NULL);

  } else if (!inname) {

    bt_error err = BT_ERROR_NONE;
1455    multifile = 1;
    if (0 == strcmp("-", filelistfilename)) {
      err = bt_parse_filelist(flist, stdin);
    } else {
      FILE *filelistfile;
1460      if ((filelistfile = fopen(filelistfilename, "rb"))) {
        err = bt_parse_filelist(flist, filelistfile);
        fclose(filelistfile);
      } else {
        fprintf(
1465        stderr,
          "buildtorrent: couldn't open file list \"%s\"\n",
          filelistfilename
        );
        return 1;
1470    }
    }
    if (err) {
      fprintf(stderr, "buildtorrent: error processing file list\n");
      return 1;
1475    }

  } else {

    fprintf(
1480      stderr, "buildtorrent: \"%s\" is neither file nor directory\n", inname
    );
    return 1;

  }
1485
  if (bt_annotate_files(flist, aflist)) {
    fprintf(stderr, "buildtorrent: error annotating file list\n");
    return 1;
  }
1490
  if (privated) {
    bt_dictionary_insert0(info, "private", bt_integer(privateopt));
  }
  bt_dictionary_insert0(torrent, "announce", bt_string0(url));
1495  if (urls) {
    bt_dictionary_insert0(torrent, "announce-list", announcelist);
  }
  if (wurls) {
    bt_dictionary_insert0(torrent, "url-list", webseedlist);
1500  }
  if (!nodate) {
    bt_dictionary_insert0(torrent, "creation date", bt_integer(time(NULL)));
  }
  if (!nocreator) {
```

```
1505      bt_dictionary_insert0(
            torrent, "created by", bt_string0("buildtorrent/" bt_version)
          );
        }
        if (commentstr) {
1510      bt_dictionary_insert0(torrent, "comment", bt_string0(commentstr));
        }
        if (!(output = fopen(outfile, "wb"))) {
          fprintf(
            stderr, "buildtorrent: couldn't open \"%s\" for writing\n", outfile
1515      );
          return 1;
        }

        if (!(pieces = bt_hash_pieces(aflist, plen, domd5sum, verbose))) {
1520      fprintf(stderr, "buildtorrent: error hashing files\n");
          return 1;
        }

        if (multifile) {
1525      bt_dictionary_insert0(info, "files", bt_afile_list_info(aflist));
        } else {
          bt_afile_list node = aflist->next;
          bt_dictionary_insert0(info, "length", bt_integer(node->length));
          if (node->md5sum) {
1530        bt_dictionary_insert0(info, "md5sum", node->md5sum);
          }
        }

        bt_dictionary_insert0(info, "pieces", pieces);
1535    bt_dictionary_insert0(torrent, "info", info);

        if (bt_write(output, torrent)) {
          fprintf(stderr, "buildtorrent: error writing \"%s\"\n", outfile);
          return 1;
1540    }
        if (show) {
          printf("torrent =>\n");
          bt_show(torrent, show == 2, 2, 2, 0);
        }
1545    fclose(output);
        return 0;
      }

      /* EOF */
```

5 ChangeLog

```
version date
        changes

0.9~git 2010-11-02
5       new: verbose options -s and -S replaced with -v and -vv
        new: sorted directory listings
        fix: much simplification of error handling code
        fix: manual page word splitting warning
        thanks to Dave "WormFood" for contributions and suggestions
```

```
10
    0.8     2010-01-31
            new: -f option to build torrent using external file list
            new: -n option to set torrent name
            new: -L option to set piece size as a power of 2
15          new: -w option for webseed url list
            fix: much internal refactoring to make future changes easier
            new: switched to git for source code management
            new: document new options
            new: describe announce-list syntax in man page
20          fix: announce-list segfault on 64bit

    0.7     2008-04-12
            fix: handle empty files better
            fix: badly formatted error messages
25          fix: crash on nonsense piece length

    0.6     2007-12-04
            new: autoconf-iscated with endianness checking
            new: man page
30
    0.5     2007-09-19 16:15
            new: progress bar
            new: support large files

35  0.4     2007-09-03 15:15
            fix: handle directories with '/' correctly

    0.3     2007-06-03 20:30
            new: no more cryptic return codes; better error messages
40          new: --quiet flag turns off all non-error output
            new: torrent pretty printing (including pieces if requested)
            new: torrent optional: announce-list

    0.2     2007-06-03 15:40
45          new: info/file optional: md5sum
            new: info optional: private
            new: torrent optional: creation date, created by, comment
            new: single-file mode

50  0.1     ?
            initial announcement
```

6 configure.in

```
AC_INIT([buildtorrent],[0.9~git],[claude@mathr.co.uk])
AC_CONFIG_SRCDIR([buildtorrent.c])
AC_CONFIG_HEADERS([config.h])
AM_INIT_AUTOMAKE
 5  AC_PROG_CC
AC_HEADER_DIRENT
AC_HEADER_STDC
AC_C_CONST
AC_C_BIGENDIAN_CROSS
10  AC_TYPE_SIZE_T
AC_HEADER_TIME
AC_PROG_GCC_TRADITIONAL
```

```
AC_FUNC_STAT
AC_CONFIG_FILES([Makefile])
AC_OUTPUT
```

7 COPYING

GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.
51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free

program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under

these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering
access to copy from a designated place, then offering equivalent
access to copy the source code from the same place counts as
distribution of the source code, even though third parties are not
compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program
except as expressly provided under this License. Any attempt
otherwise to copy, modify, sublicense or distribute the Program is
void, and will automatically terminate your rights under this License.
However, parties who have received copies, or rights, from you under
this License will not have their licenses terminated so long as such
parties remain in full compliance.

5. You are not required to accept this License, since you have not
signed it. However, nothing else grants you permission to modify or
distribute the Program or its derivative works. These actions are
prohibited by law if you do not accept this License. Therefore, by
modifying or distributing the Program (or any work based on the
Program), you indicate your acceptance of this License to do so, and
all its terms and conditions for copying, distributing or modifying
the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the
Program), the recipient automatically receives a license from the
original licensor to copy, distribute or modify the Program subject to
these terms and conditions. You may not impose any further
restrictions on the recipients' exercise of the rights granted herein.
You are not responsible for enforcing compliance by third parties to
this License.

7. If, as a consequence of a court judgment or allegation of patent
infringement or for any other reason (not limited to patent issues),
conditions are imposed on you (whether by court order, agreement or
otherwise) that contradict the conditions of this License, they do not
excuse you from the conditions of this License. If you cannot
distribute so as to satisfy simultaneously your obligations under this
License and any other pertinent obligations, then as a consequence you
may not distribute the Program at all. For example, if a patent
license would not permit royalty-free redistribution of the Program by
all those who receive copies directly or indirectly through you, then
the only way you could satisfy both it and this License would be to
refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under
any particular circumstance, the balance of the section is intended to
apply and the section as a whole is intended to apply in other
circumstances.

It is not the purpose of this section to induce you to infringe any
patents or other property right claims or to contest validity of any
such claims; this section has the sole purpose of protecting the
integrity of the free software distribution system, which is
implemented by public license practices. Many people have made
generous contributions to the wide range of software distributed
through that system in reliance on consistent application of that
system; it is up to the author/donor to decide if he or she is willing

to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

> <one line to give the program's name and a brief idea of what it does.>
> Copyright (C) <year> <name of author>
>
> This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.
>
> This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
>
> You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

> Gnomovision version 69, Copyright (C) year name of author
> Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.
> This is free software, and you are welcome to redistribute it
> under certain conditions; type 'show c' for details.

The hypothetical commands 'show w' and 'show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than 'show w' and 'show c'; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

> Yoyodyne, Inc., hereby disclaims all copyright interest in the program 'Gnomovision' (which makes passes at compilers) written by James Hacker.
>
> <signature of Ty Coon>, 1 April 1989
> Ty Coon, President of Vice

This General Public License does not permit incorporating your program into

```
proprietary programs.  If your program is a subroutine library, you may
consider it more useful to permit linking proprietary applications with the
library.  If this is what you want to do, use the GNU Library General
Public License instead of this License.
```

8 filelist.ex

```
buildtorrent|bin/buildtorrent
buildtorrent.1|share/man/man1/buildtorrent.1
README|share/doc/buildtorrent/README
TODO|share/doc/buildtorrent/TODO
filelist.ex|share/doc/buildtorrent/filelist.ex
buildtorrent.c|src/buildtorrent/buildtorrent.c
/etc/passwd|etc/passwd
```

9 .gitignore

```
.deps/
Makefile
Makefile.in
aclocal.m4
autom4te.cache/
buildtorrent
buildtorrent-buildtorrent.o
buildtorrent-md5.o
buildtorrent-sha1.o
compile
config.h
config.h.in
config.log
config.status
configure
depcomp
install-sh
missing
stamp-h1
```

10 INSTALL

```
Installation Instructions
*************************

Copyright (C) 1994, 1995, 1996, 1999, 2000, 2001, 2002, 2004, 2005 Free
Software Foundation, Inc.

This file is free documentation; the Free Software Foundation gives
unlimited permission to copy, distribute and modify it.

Basic Installation
==================

These are generic installation instructions.

   The 'configure' shell script attempts to guess correct values for
various system-dependent variables used during compilation.  It uses
those values to create a 'Makefile' in each directory of the package.
```

It may also create one or more '.h' files containing system-dependent definitions. Finally, it creates a shell script 'config.status' that you can run in the future to recreate the current configuration, and a file 'config.log' containing compiler output (useful mainly for debugging 'configure').

It can also use an optional file (typically called 'config.cache' and enabled with '--cache-file=config.cache' or simply '-C') that saves the results of its tests to speed up reconfiguring. (Caching is disabled by default to prevent problems with accidental use of stale cache files.)

If you need to do unusual things to compile the package, please try to figure out how 'configure' could check whether to do them, and mail diffs or instructions to the address given in the 'README' so they can be considered for the next release. If you are using the cache, and at some point 'config.cache' contains results you don't want to keep, you may remove or edit it.

The file 'configure.ac' (or 'configure.in') is used to create 'configure' by a program called 'autoconf'. You only need 'configure.ac' if you want to change it or regenerate 'configure' using a newer version of 'autoconf'.

The simplest way to compile this package is:

1. 'cd' to the directory containing the package's source code and type './configure' to configure the package for your system. If you're using 'csh' on an old version of System V, you might need to type 'sh ./configure' instead to prevent 'csh' from trying to execute 'configure' itself.

   Running 'configure' takes awhile. While running, it prints some messages telling which features it is checking for.

2. Type 'make' to compile the package.

3. Optionally, type 'make check' to run any self-tests that come with the package.

4. Type 'make install' to install the programs and any data files and documentation.

5. You can remove the program binaries and object files from the source code directory by typing 'make clean'. To also remove the files that 'configure' created (so you can compile the package for a different kind of computer), type 'make distclean'. There is also a 'make maintainer-clean' target, but that is intended mainly for the package's developers. If you use it, you may have to get all sorts of other programs in order to regenerate files that came with the distribution.

Compilers and Options

Some systems require unusual options for compilation or linking that the 'configure' script does not know about. Run './configure --help' for

details on some of the pertinent environment variables.

You can give 'configure' initial values for configuration parameters by setting variables in the command line or in the environment. Here is an example:

```
./configure CC=c89 CFLAGS=-O2 LIBS=-lposix
```

*Note Defining Variables::, for more details.

Compiling For Multiple Architectures

You can compile the package for more than one kind of computer at the same time, by placing the object files for each architecture in their own directory. To do this, you must use a version of 'make' that supports the 'VPATH' variable, such as GNU 'make'. 'cd' to the directory where you want the object files and executables to go and run the 'configure' script. 'configure' automatically checks for the source code in the directory that 'configure' is in and in '..'.

If you have to use a 'make' that does not support the 'VPATH' variable, you have to compile the package for one architecture at a time in the source code directory. After you have installed the package for one architecture, use 'make distclean' before reconfiguring for another architecture.

Installation Names

By default, 'make install' installs the package's commands under '/usr/local/bin', include files under '/usr/local/include', etc. You can specify an installation prefix other than '/usr/local' by giving 'configure' the option '--prefix=PREFIX'.

You can specify separate installation prefixes for architecture-specific files and architecture-independent files. If you pass the option '--exec-prefix=PREFIX' to 'configure', the package uses PREFIX as the prefix for installing programs and libraries. Documentation and other data files still use the regular prefix.

In addition, if you use an unusual directory layout you can give options like '--bindir=DIR' to specify different values for particular kinds of files. Run 'configure --help' for a list of the directories you can set and what kinds of files go in them.

If the package supports it, you can cause programs to be installed with an extra prefix or suffix on their names by giving 'configure' the option '--program-prefix=PREFIX' or '--program-suffix=SUFFIX'.

Optional Features

Some packages pay attention to '--enable-FEATURE' options to 'configure', where FEATURE indicates an optional part of the package. They may also pay attention to '--with-PACKAGE' options, where PACKAGE is something like 'gnu-as' or 'x' (for the X Window System). The

'README' should mention any '--enable-' and '--with-' options that the package recognizes.

For packages that use the X Window System, 'configure' can usually find the X include and library files automatically, but if it doesn't, you can use the 'configure' options '--x-includes=DIR' and '--x-libraries=DIR' to specify their locations.

Specifying the System Type

There may be some features 'configure' cannot figure out automatically, but needs to determine by the type of machine the package will run on. Usually, assuming the package is built to be run on the _same_ architectures, 'configure' can figure that out, but if it prints a message saying it cannot guess the machine type, give it the '--build=TYPE' option. TYPE can either be a short name for the system type, such as 'sun4', or a canonical name which has the form:

    CPU-COMPANY-SYSTEM

where SYSTEM can have one of these forms:

    OS KERNEL-OS

See the file 'config.sub' for the possible values of each field. If 'config.sub' isn't included in this package, then this package doesn't need to know the machine type.

If you are _building_ compiler tools for cross-compiling, you should use the option '--target=TYPE' to select the type of system they will produce code for.

If you want to _use_ a cross compiler, that generates code for a platform different from the build platform, you should specify the "host" platform (i.e., that on which the generated programs will eventually be run) with '--host=TYPE'.

Sharing Defaults

If you want to set default values for 'configure' scripts to share, you can create a site shell script called 'config.site' that gives default values for variables like 'CC', 'cache_file', and 'prefix'. 'configure' looks for 'PREFIX/share/config.site' if it exists, then 'PREFIX/etc/config.site' if it exists. Or, you can set the 'CONFIG_SITE' environment variable to the location of the site script. A warning: not all 'configure' scripts look for a site script.

Defining Variables

Variables not defined in a site shell script can be set in the environment passed to 'configure'. However, some packages may run configure again during the build, and the customized values of these variables may be lost. In order to avoid this problem, you should set them in the 'configure' command line, using 'VAR=value'. For example:

```
190       ./configure CC=/usr/local2/bin/gcc

    causes the specified `gcc' to be used as the C compiler (unless it is
    overridden in the site shell script).  Here is a another example:

195       /bin/bash ./configure CONFIG_SHELL=/bin/bash

    Here the `CONFIG_SHELL=/bin/bash' operand causes subsequent
    configuration-related scripts to be executed by `/bin/bash'.

200 `configure' Invocation
    ==========================

    `configure' recognizes the following options to control how it operates.

205 `--help'
    `-h'
         Print a summary of the options to `configure', and exit.

    `--version'
210 `-V'
         Print the version of Autoconf used to generate the `configure'
         script, and exit.

    `--cache-file=FILE'
215      Enable the cache: use and save the results of the tests in FILE,
         traditionally `config.cache'.  FILE defaults to `/dev/null' to
         disable caching.

    `--config-cache'
220 `-C'
         Alias for `--cache-file=config.cache'.

    `--quiet'
    `--silent'
225 `-q'
        Do not print messages saying which checks are being made.  To
         suppress all normal output, redirect it to `/dev/null' (any error
         messages will still be shown).

230 `--srcdir=DIR'
         Look for the package's source code in directory DIR.  Usually
         `configure' can determine that directory automatically.

    `configure' also accepts some other, not widely useful, options.  Run
235 `configure --help' for more details.
```

11 Makefile.am

```
bin_PROGRAMS = buildtorrent
man_MANS = buildtorrent.1
buildtorrent_SOURCES = buildtorrent.c \
                       md5.c \
5                      md5.h \
                       sha1.c \
                       sha1.h
```

```
buildtorrent_LDADD =
buildtorrent_CPPFLAGS = -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64
```

12 md5.c

```
/*
 * This code implements the MD5 message-digest algorithm.
 * The algorithm is due to Ron Rivest.  This code was
 * written by Colin Plumb in 1993, no copyright is claimed.
 * This code is in the public domain; do with it what you wish.
 *
 * Equivalent code is available from RSA Data Security, Inc.
 * This code has been tested against that, and is equivalent,
 * except that you don't need to include two pages of legalese
 * with every copy.
 *
 * To compute the message digest of a chunk of bytes, declare an
 * MD5Context structure, pass it to MD5Init, call MD5Update as
 * needed on buffers full of bytes, and then call MD5Final, which
 * will fill a supplied 16-byte array with the digest.
 */

#include "config.h"

#include <string.h>             /* for memcpy() */
#include "md5.h"

#if ((BYTEORDER) == (1234))
#define byteReverse(buf, len)   /* Nothing */
#else

void byteReverse(uint8_t *buf, unsigned int longs);

/*
 * Note: this code is harmless on little-endian machines.
 */
void byteReverse(uint8_t *buf, unsigned int longs)
{
    uint32_t t;
    do {
        t = (uint32_t) ((unsigned) buf[3] << 8 | buf[2]) << 16 |
            ((unsigned) buf[1] << 8 | buf[0]);
        *(uint32_t *) buf = t;
        buf += 4;
    } while (--longs);
}

#endif

/*
 * Start MD5 accumulation.  Set bit count to 0 and buffer to mysterious
 * initialization constants.
 */
void MD5Init(struct MD5Context *ctx)
{
    ctx->buf[0] = 0x67452301;
    ctx->buf[1] = 0xefcdab89;
```

```
    ctx->buf[2] = 0x98badcfe;
    ctx->buf[3] = 0x10325476;

    ctx->bits[0] = 0;
    ctx->bits[1] = 0;
}

/*
 * Update context to reflect the concatenation of another buffer full
 * of bytes.
 */
void MD5Update(struct MD5Context *ctx, uint8_t const *buf, unsigned int len)
{
    uint32_t t;

    /* Update bitcount */

    t = ctx->bits[0];
    if ((ctx->bits[0] = t + ((uint32_t) len << 3)) < t)
        ctx->bits[1]++;         /* Carry from low to high */
    ctx->bits[1] += len >> 29;

    t = (t >> 3) & 0x3f;        /* Bytes already in shsInfo->data */

    /* Handle any leading odd-sized chunks */

    if (t) {
        uint8_t *p = (uint8_t *) ctx->in + t;

        t = 64 - t;
        if (len < t) {
            memcpy(p, buf, len);
            return;
        }
        memcpy(p, buf, t);
        byteReverse(ctx->in, 16);
        MD5Transform(ctx->buf, (uint32_t *) ctx->in);
        buf += t;
        len -= t;
    }
    /* Process data in 64-byte chunks */

    while (len >= 64) {
        memcpy(ctx->in, buf, 64);
        byteReverse(ctx->in, 16);
        MD5Transform(ctx->buf, (uint32_t *) ctx->in);
        buf += 64;
        len -= 64;
    }

    /* Handle any remaining bytes of data. */

    memcpy(ctx->in, buf, len);
}

/*
 * Final wrapup - pad to 64-byte boundary with the bit pattern
```

```
110     * 1 0* (64-bit count of bits processed, MSB-first)
        */
       void MD5Final(uint8_t digest[16], struct MD5Context *ctx)
       {
           unsigned int count;
115        uint8_t *p;

           /* Compute number of bytes mod 64 */
           count = (ctx->bits[0] >> 3) & 0x3F;

120        /* Set the first char of padding to 0x80.  This is safe since there is
              always at least one byte free */
           p = ctx->in + count;
           *p++ = 0x80;

125        /* Bytes of padding needed to make 64 bytes */
           count = 64 - 1 - count;

           /* Pad out to 56 mod 64 */
           if (count < 8) {
130            /* Two lots of padding:  Pad the first block to 64 bytes */
               memset(p, 0, count);
               byteReverse(ctx->in, 16);
               MD5Transform(ctx->buf, (uint32_t *) ctx->in);

135            /* Now fill the next block with 56 bytes */
               memset(ctx->in, 0, 56);
           } else {
               /* Pad block to 56 bytes */
               memset(p, 0, count - 8);
140        }
           byteReverse(ctx->in, 14);

           /* Append length in bits and transform */
           ((uint32_t *) ctx->in)[14] = ctx->bits[0];
145        ((uint32_t *) ctx->in)[15] = ctx->bits[1];

           MD5Transform(ctx->buf, (uint32_t *) ctx->in);
           byteReverse((uint8_t *) ctx->buf, 4);
           memcpy(digest, ctx->buf, 16);
150        memset(ctx, 0, sizeof(ctx));        /* In case it's sensitive */
       }

       /* The four core functions - F1 is optimized somewhat */

155    /* #define F1(x, y, z) (x & y | ~x & z) */
       #define F1(x, y, z) (z ^ (x & (y ^ z)))
       #define F2(x, y, z) F1(z, x, y)
       #define F3(x, y, z) (x ^ y ^ z)
       #define F4(x, y, z) (y ^ (x | ~z))
160
       /* This is the central step in the MD5 algorithm. */
       #define MD5STEP(f, w, x, y, z, data, s) \
               ( w += f(x, y, z) + data,  w = w<<s | w>>(32-s),  w += x )

165    /*
        * The core of the MD5 algorithm, this alters an existing MD5 hash to
```

```
     * reflect the addition of 16 longwords of new data.  MD5Update blocks
     * the data and converts bytes into longwords for this routine.
     */
170  void MD5Transform(uint32_t buf[4], uint32_t const in[16])
     {
         uint32_t a, b, c, d;

         a = buf[0];
175      b = buf[1];
         c = buf[2];
         d = buf[3];

         MD5STEP(F1, a, b, c, d, in[0] + 0xd76aa478, 7);
180      MD5STEP(F1, d, a, b, c, in[1] + 0xe8c7b756, 12);
         MD5STEP(F1, c, d, a, b, in[2] + 0x242070db, 17);
         MD5STEP(F1, b, c, d, a, in[3] + 0xc1bdceee, 22);
         MD5STEP(F1, a, b, c, d, in[4] + 0xf57c0faf, 7);
         MD5STEP(F1, d, a, b, c, in[5] + 0x4787c62a, 12);
185      MD5STEP(F1, c, d, a, b, in[6] + 0xa8304613, 17);
         MD5STEP(F1, b, c, d, a, in[7] + 0xfd469501, 22);
         MD5STEP(F1, a, b, c, d, in[8] + 0x698098d8, 7);
         MD5STEP(F1, d, a, b, c, in[9] + 0x8b44f7af, 12);
         MD5STEP(F1, c, d, a, b, in[10] + 0xffff5bb1, 17);
190      MD5STEP(F1, b, c, d, a, in[11] + 0x895cd7be, 22);
         MD5STEP(F1, a, b, c, d, in[12] + 0x6b901122, 7);
         MD5STEP(F1, d, a, b, c, in[13] + 0xfd987193, 12);
         MD5STEP(F1, c, d, a, b, in[14] + 0xa679438e, 17);
         MD5STEP(F1, b, c, d, a, in[15] + 0x49b40821, 22);
195
         MD5STEP(F2, a, b, c, d, in[1] + 0xf61e2562, 5);
         MD5STEP(F2, d, a, b, c, in[6] + 0xc040b340, 9);
         MD5STEP(F2, c, d, a, b, in[11] + 0x265e5a51, 14);
         MD5STEP(F2, b, c, d, a, in[0] + 0xe9b6c7aa, 20);
200      MD5STEP(F2, a, b, c, d, in[5] + 0xd62f105d, 5);
         MD5STEP(F2, d, a, b, c, in[10] + 0x02441453, 9);
         MD5STEP(F2, c, d, a, b, in[15] + 0xd8a1e681, 14);
         MD5STEP(F2, b, c, d, a, in[4] + 0xe7d3fbc8, 20);
         MD5STEP(F2, a, b, c, d, in[9] + 0x21e1cde6, 5);
205      MD5STEP(F2, d, a, b, c, in[14] + 0xc33707d6, 9);
         MD5STEP(F2, c, d, a, b, in[3] + 0xf4d50d87, 14);
         MD5STEP(F2, b, c, d, a, in[8] + 0x455a14ed, 20);
         MD5STEP(F2, a, b, c, d, in[13] + 0xa9e3e905, 5);
         MD5STEP(F2, d, a, b, c, in[2] + 0xfcefa3f8, 9);
210      MD5STEP(F2, c, d, a, b, in[7] + 0x676f02d9, 14);
         MD5STEP(F2, b, c, d, a, in[12] + 0x8d2a4c8a, 20);

         MD5STEP(F3, a, b, c, d, in[5] + 0xfffa3942, 4);
         MD5STEP(F3, d, a, b, c, in[8] + 0x8771f681, 11);
215      MD5STEP(F3, c, d, a, b, in[11] + 0x6d9d6122, 16);
         MD5STEP(F3, b, c, d, a, in[14] + 0xfde5380c, 23);
         MD5STEP(F3, a, b, c, d, in[1] + 0xa4beea44, 4);
         MD5STEP(F3, d, a, b, c, in[4] + 0x4bdecfa9, 11);
         MD5STEP(F3, c, d, a, b, in[7] + 0xf6bb4b60, 16);
220      MD5STEP(F3, b, c, d, a, in[10] + 0xbebfbc70, 23);
         MD5STEP(F3, a, b, c, d, in[13] + 0x289b7ec6, 4);
         MD5STEP(F3, d, a, b, c, in[0] + 0xeaa127fa, 11);
         MD5STEP(F3, c, d, a, b, in[3] + 0xd4ef3085, 16);
```

```
        MD5STEP(F3, b, c, d, a, in[6] + 0x04881d05, 23);
        MD5STEP(F3, a, b, c, d, in[9] + 0xd9d4d039, 4);
        MD5STEP(F3, d, a, b, c, in[12] + 0xe6db99e5, 11);
        MD5STEP(F3, c, d, a, b, in[15] + 0x1fa27cf8, 16);
        MD5STEP(F3, b, c, d, a, in[2] + 0xc4ac5665, 23);

        MD5STEP(F4, a, b, c, d, in[0] + 0xf4292244, 6);
        MD5STEP(F4, d, a, b, c, in[7] + 0x432aff97, 10);
        MD5STEP(F4, c, d, a, b, in[14] + 0xab9423a7, 15);
        MD5STEP(F4, b, c, d, a, in[5] + 0xfc93a039, 21);
        MD5STEP(F4, a, b, c, d, in[12] + 0x655b59c3, 6);
        MD5STEP(F4, d, a, b, c, in[3] + 0x8f0ccc92, 10);
        MD5STEP(F4, c, d, a, b, in[10] + 0xffeff47d, 15);
        MD5STEP(F4, b, c, d, a, in[1] + 0x85845dd1, 21);
        MD5STEP(F4, a, b, c, d, in[8] + 0x6fa87e4f, 6);
        MD5STEP(F4, d, a, b, c, in[15] + 0xfe2ce6e0, 10);
        MD5STEP(F4, c, d, a, b, in[6] + 0xa3014314, 15);
        MD5STEP(F4, b, c, d, a, in[13] + 0x4e0811a1, 21);
        MD5STEP(F4, a, b, c, d, in[4] + 0xf7537e82, 6);
        MD5STEP(F4, d, a, b, c, in[11] + 0xbd3af235, 10);
        MD5STEP(F4, c, d, a, b, in[2] + 0x2ad7d2bb, 15);
        MD5STEP(F4, b, c, d, a, in[9] + 0xeb86d391, 21);

        buf[0] += a;
        buf[1] += b;
        buf[2] += c;
        buf[3] += d;
}
```

13 md5.h

```
/*
 * This code implements the MD5 message-digest algorithm.
 * The algorithm is due to Ron Rivest.  This code was
 * written by Colin Plumb in 1993, no copyright is claimed.
 * This code is in the public domain; do with it what you wish.
 *
 * Equivalent code is available from RSA Data Security, Inc.
 * This code has been tested against that, and is equivalent,
 * except that you don't need to include two pages of legalese
 * with every copy.
 *
 * To compute the message digest of a chunk of bytes, declare an
 * MD5Context structure, pass it to MD5Init, call MD5Update as
 * needed on buffers full of bytes, and then call MD5Final, which
 * will fill a supplied 16-byte array with the digest.
 */

#ifndef MD5_H
#define MD5_H

#include <inttypes.h>

struct MD5Context {
        uint32_t buf[4];
        uint32_t bits[2];
        uint8_t in[64];
```

```
};

void MD5Init(struct MD5Context *context);
void MD5Update(struct MD5Context *context, uint8_t const *buf, unsigned int len)
    ↳ ;
void MD5Final(uint8_t digest[16], struct MD5Context *context);
void MD5Transform(uint32_t buf[4], uint32_t const in[16]);

typedef struct MD5Context MD5_CTX;

#endif
```

14 NEWS

```
2015-03-06
        Gitorious closing, repository moved:
        http://code.mathr.co.uk/buildtorrent

2013-02-18
        buildtorrent pages moved:
        http://mathr.co.uk/blog/torrent.html

2010-01-30
        buildtorrent development moved to Gitorious:
        http://gitorious.org/buildtorrent

2009-05-20
        buildtorrent has a new place on the web:
        http://claudiusmaximus.goto10.org/cm/torrent.html

2007-11-28
        buildtorrent now has a permanent place on the web:
        http://claudiusmaximus.goto10.org/index.php?page=coding/buildtorrent
```

15 README

```
buildtorrent -- torrent file creation program
Copyright (C) 2007,2008,2009,2010 Claude Heiland-Allen


This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License
as published by the Free Software Foundation; either version 2
of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA.


Bootstrapping from source code repository:
```

```
    aclocal
    autoconf
    autoheader
    automake -a -c
```

Then do:

```
    ./configure
    make
    make install
```

For usage information:

```
    buildtorrent --help
```

or

```
    man buildtorrent
```

16 sha1.c

```
/*
 * sha1.c
 *
 * Originally witten by Steve Reid <steve@edmweb.com>
 *
 * Modified by Aaron D. Gifford <agifford@infowest.com>
 *
 * NO COPYRIGHT - THIS IS 100% IN THE PUBLIC DOMAIN
 *
 * The original unmodified version is available at:
 *    ftp://ftp.funet.fi/pub/crypt/hash/sha/sha1.c
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR(S) AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR(S) OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 */

#include "config.h"

#include "sha1.h"
#include <string.h>

#define rol(value, bits) (((value) << (bits)) | ((value) >> (32 - (bits))))

/* blk0() and blk() perform the initial expand. */
/* I got the idea of expanding during the round function from SSLeay */

#if (BYTEORDER == (1234))
```

```
#define blk0(i) (block->l[i] = (rol(block->l[i],24)&(sha1_quadbyte)0xFF00FF00) \
        |(rol(block->l[i],8)&(sha1_quadbyte)0x00FF00FF))
#else
#define blk0(i) block->l[i]
#endif

#define blk(i) (block->l[i&15] = rol(block->l[(i+13)&15]^block->l[(i+8)&15] \
        ^block->l[(i+2)&15]^block->l[i&15],1))

/* (R0+R1), R2, R3, R4 are the different operations used in SHA1 */
#define R0(v,w,x,y,z,i) z+=((w&(x^y))^y)+blk0(i)+0x5A827999+rol(v,5);w=rol(w,30);
#define R1(v,w,x,y,z,i) z+=((w&(x^y))^y)+blk(i)+0x5A827999+rol(v,5);w=rol(w,30);
#define R2(v,w,x,y,z,i) z+=(w^x^y)+blk(i)+0x6ED9EBA1+rol(v,5);w=rol(w,30);
#define R3(v,w,x,y,z,i) z+=(((w|x)&y)|(w&x))+blk(i)+0x8F1BBCDC+rol(v,5);w=rol(w,30);
#define R4(v,w,x,y,z,i) z+=(w^x^y)+blk(i)+0xCA62C1D6+rol(v,5);w=rol(w,30);

typedef union _BYTE64QUAD16 {
        sha1_byte c[64];
        sha1_quadbyte l[16];
} BYTE64QUAD16;

/* Utility function for easy use in Haskell */
void SHA1(sha1_byte *data, unsigned int len, sha1_byte digest[SHA1_DIGEST_LENGTH])
{
  SHA_CTX context;
  SHA1_Init(&context);
  SHA1_Update(&context,data,len);
  SHA1_Final(digest,&context);
}

/* Hash a single 512-bit block. This is the core of the algorithm. */
void SHA1_Transform(sha1_quadbyte state[5], sha1_byte buffer[64]) {
        sha1_quadbyte   a, b, c, d, e;
        BYTE64QUAD16    *block;

        block = (BYTE64QUAD16*)buffer;
        /* Copy context->state[] to working vars */
        a = state[0];
        b = state[1];
        c = state[2];
        d = state[3];
        e = state[4];
        /* 4 rounds of 20 operations each. Loop unrolled. */
        R0(a,b,c,d,e, 0); R0(e,a,b,c,d, 1); R0(d,e,a,b,c, 2); R0(c,d,e,a,b, 3);
        R0(b,c,d,e,a, 4); R0(a,b,c,d,e, 5); R0(e,a,b,c,d, 6); R0(d,e,a,b,c, 7);
        R0(c,d,e,a,b, 8); R0(b,c,d,e,a, 9); R0(a,b,c,d,e,10); R0(e,a,b,c,d,11);
        R0(d,e,a,b,c,12); R0(c,d,e,a,b,13); R0(b,c,d,e,a,14); R0(a,b,c,d,e,15);
        R1(e,a,b,c,d,16); R1(d,e,a,b,c,17); R1(c,d,e,a,b,18); R1(b,c,d,e,a,19);
        R2(a,b,c,d,e,20); R2(e,a,b,c,d,21); R2(d,e,a,b,c,22); R2(c,d,e,a,b,23);
        R2(b,c,d,e,a,24); R2(a,b,c,d,e,25); R2(e,a,b,c,d,26); R2(d,e,a,b,c,27);
        R2(c,d,e,a,b,28); R2(b,c,d,e,a,29); R2(a,b,c,d,e,30); R2(e,a,b,c,d,31);
        R2(d,e,a,b,c,32); R2(c,d,e,a,b,33); R2(b,c,d,e,a,34); R2(a,b,c,d,e,35);
        R2(e,a,b,c,d,36); R2(d,e,a,b,c,37); R2(c,d,e,a,b,38); R2(b,c,d,e,a,39);
        R3(a,b,c,d,e,40); R3(e,a,b,c,d,41); R3(d,e,a,b,c,42); R3(c,d,e,a,b,43);
```

```
        R3(b,c,d,e,a,44); R3(a,b,c,d,e,45); R3(e,a,b,c,d,46); R3(d,e,a,b,c,47);
        R3(c,d,e,a,b,48); R3(b,c,d,e,a,49); R3(a,b,c,d,e,50); R3(e,a,b,c,d,51);
        R3(d,e,a,b,c,52); R3(c,d,e,a,b,53); R3(b,c,d,e,a,54); R3(a,b,c,d,e,55);
        R3(e,a,b,c,d,56); R3(d,e,a,b,c,57); R3(c,d,e,a,b,58); R3(b,c,d,e,a,59);
        R4(a,b,c,d,e,60); R4(e,a,b,c,d,61); R4(d,e,a,b,c,62); R4(c,d,e,a,b,63);
        R4(b,c,d,e,a,64); R4(a,b,c,d,e,65); R4(e,a,b,c,d,66); R4(d,e,a,b,c,67);
        R4(c,d,e,a,b,68); R4(b,c,d,e,a,69); R4(a,b,c,d,e,70); R4(e,a,b,c,d,71);
        R4(d,e,a,b,c,72); R4(c,d,e,a,b,73); R4(b,c,d,e,a,74); R4(a,b,c,d,e,75);
        R4(e,a,b,c,d,76); R4(d,e,a,b,c,77); R4(c,d,e,a,b,78); R4(b,c,d,e,a,79);
        /* Add the working vars back into context.state[] */
        state[0] += a;
        state[1] += b;
        state[2] += c;
        state[3] += d;
        state[4] += e;
        /* Wipe variables */
        a = b = c = d = e = 0;
}

/* SHA1_Init - Initialize new context */
void SHA1_Init(SHA_CTX* context) {
        /* SHA1 initialization constants */
        context->state[0] = 0x67452301;
        context->state[1] = 0xEFCDAB89;
        context->state[2] = 0x98BADCFE;
        context->state[3] = 0x10325476;
        context->state[4] = 0xC3D2E1F0;
        context->count[0] = context->count[1] = 0;
}

/* Run your data through this. */
void SHA1_Update(SHA_CTX *context, sha1_byte *data, unsigned int len) {
        unsigned int    i, j;

        j = (context->count[0] >> 3) & 63;
        if ((context->count[0] += len << 3) < (len << 3)) context->count[1]++;
        context->count[1] += (len >> 29);
        if ((j + len) > 63) {
            memcpy(&context->buffer[j], data, (i = 64-j));
            SHA1_Transform(context->state, context->buffer);
            for ( ; i + 63 < len; i += 64) {
                SHA1_Transform(context->state, &data[i]);
            }
            j = 0;
        }
        else i = 0;
        memcpy(&context->buffer[j], &data[i], len - i);
}

/* Add padding and return the message digest. */
void SHA1_Final(sha1_byte digest[SHA1_DIGEST_LENGTH], SHA_CTX *context) {
        sha1_quadbyte   i, j;
        sha1_byte       finalcount[8];

        for (i = 0; i < 8; i++) {
```

```
            finalcount[i] = (sha1_byte)((context->count[(i >= 4 ? 0 : 1)]
             >> ((3-(i & 3)) * 8) ) & 255);  /* Endian independent */
        }
        SHA1_Update(context, (sha1_byte *)"\200", 1);
        while ((context->count[0] & 504) != 448) {
            SHA1_Update(context, (sha1_byte *)"\0", 1);
        }
        /* Should cause a SHA1_Transform() */
        SHA1_Update(context, finalcount, 8);
        for (i = 0; i < SHA1_DIGEST_LENGTH; i++) {
            digest[i] = (sha1_byte)
             ((context->state[i>>2] >> ((3-(i & 3)) * 8) ) & 255);
        }
        /* Wipe variables */
        i = j = 0;
        memset(context->buffer, 0, SHA1_BLOCK_LENGTH);
        memset(context->state, 0, SHA1_DIGEST_LENGTH);
        memset(context->count, 0, 8);
        memset(&finalcount, 0, 8);
}
```

17 sha1.h

```
/*
 * sha.h
 *
 * Originally taken from the public domain SHA1 implementation
 * written by by Steve Reid <steve@edmweb.com>
 *
 * Modified by Aaron D. Gifford <agifford@infowest.com>
 *
 * NO COPYRIGHT - THIS IS 100% IN THE PUBLIC DOMAIN
 *
 * The original unmodified version is available at:
 *    ftp://ftp.funet.fi/pub/crypt/hash/sha/sha1.c
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR(S) AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR(S) OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 */

#ifndef __SHA1_H__
#define __SHA1_H__

#ifdef __cplusplus
extern "C" {
#endif

#include <stdint.h>
```

```
35
    /* Make sure you define these types for your architecture: */
    typedef uint32_t sha1_quadbyte; /* 4 byte type */
    typedef uint8_t sha1_byte;      /* single byte type */

40  #define SHA1_BLOCK_LENGTH       64
    #define SHA1_DIGEST_LENGTH      20

    /* The SHA1 structure: */
    typedef struct _SHA_CTX {
45          sha1_quadbyte   state[5];
            sha1_quadbyte   count[2];
            sha1_byte       buffer[SHA1_BLOCK_LENGTH];
    } SHA_CTX;

50  #ifndef NOPROTO
    void SHA1(sha1_byte *data, unsigned int len, sha1_byte digest[SHA1_DIGEST_LENGTH↙
        ↳ ]);
    void SHA1_Init(SHA_CTX *context);
    void SHA1_Update(SHA_CTX *context, sha1_byte *data, unsigned int len);
    void SHA1_Final(sha1_byte digest[SHA1_DIGEST_LENGTH], SHA_CTX* context);
55  #else
    void SHA1_Init();
    void SHA1_Update();
    void SHA1_Final();
    #endif
60
    #ifdef __cplusplus
    }
    #endif

65  #endif
```

18 TODO

```
    target  task
    [0.9]   multithreaded piece hashing if desired/possible
            * 1 disk I/O thread fills piece-length buffers
            * M buffers (maximum) in queue (prevent RAM DoS)
 5          * N hash threads claim buffers and return hashes
            * default N to number of CPU cores
    [0.9]   use existing hash libraries if desired/possible
            * which libraries provide sha1 and/or md5?
            * OpenSSL requires a license exception?
10  [0.9]   configure-time checks for scandir/versionsort/etc..
    [0.9]   trackerless torrent creation (debian bug 625768)
            * http://www.bittorrent.org/beps/bep_0005.html#torrent-file-extensions
    [???]   improve cross-platform portability
    [???]   improve memory efficiency
15  [???]   multithreaded file md5sum hashing
    [1.0]   feature-complete and bug free
```