```
**********************************************************
   31122 Mon Aug 27 12:18:35 2012
new/usr/src/tools/cscope-fast/invlib.c
2944 cscope-fast/invlib.[ch] use reserved word as identifier
Reviewed by: Albert Lee <trisk@nexenta.com>
Reviewed by: Gary Mills <gary_mills@fastmail.fm>
Reviewed by: Joerg Schilling <Joerg.Schilling@fokus.fraunhofer.de>
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License, Version 1.0 only
 * (the "License").  You may not use this file except in compliance
 * with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*      Copyright (c) 1988 AT&T */
/*        All Rights Reserved   */


/*
 * Copyright 2004 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

#pragma ident   "%Z%%M% %I%     %E% SMI"

#include <ctype.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/sysmacros.h>
#include <sys/byteorder.h>
#if SHARE
#include <sys/ipc.h>
#include <sys/shm.h>
#define ERR  -1
#endif
#include "invlib.h"
#include "library.h"

#define DEBUG           0       /* debugging code and realloc messages */
#define BLOCKSIZE       2 * BUFSIZ      /* logical block size */
#define LINEMAX         1000    /* sorted posting line max size */
#define POSTINC         10000   /* posting buffer size increment */
#define SEP             ' '     /* sorted posting field separator */
#define SETINC          100     /* posting set size increment */
#define STATS           0       /* print statistics */
#define SUPERINC        10000   /* super index size increment */
#define TERMMAX         512     /* term max size */
#define VERSION         1       /* inverted index format version */
#define ZIPFSIZE        200     /* zipf curve size */
#define FREAD   "r"             /* fopen for reading */
#define FREADP  "r+"            /* fopen for update */
```



```
#define FWRITE  "w"             /* fopen truncate or create for writing */
#define FWRITEP "w+"            /* fopen truncate or create for update */

extern  char    *argv0; /* command name (must be set in main function) */

int     invbreak;

#if STATS
int     showzipf;       /* show postings per term distribution */
#endif

static  POSTING *item, *enditem, *item1 = NULL, *item2 = NULL;
static  unsigned setsize1, setsize2;
static  long    numitems, totterm, zerolong;
static  char    *indexfile, *postingfile;
static  FILE    *outfile, *fpost;
static  unsigned supersize = SUPERINC, supintsize;
static  int     numpost, numlogblk, amtused, nextpost,
                lastinblk, numinvitems;
static  POSTING *POST, *postptr;
static  unsigned long   *SUPINT, *supint, nextsupfing;
static  char    *SUPFING, *supfing;
static  char    thisterm[TERMMAX];
static  union {
        long    invblk[BLOCKSIZE / sizeof (long)];
        char    chrblk[BLOCKSIZE];
} logicalblk;
_____unchanged_portion_omitted_

POSTING *
boolfile(INVCONTROL *invcntl, long *num, int op)
boolfile(INVCONTROL *invcntl, long *num, int bool)
{
        ENTRY   *entryptr;
        FILE    *file;
        char    *ptr;
        unsigned long   *ptr2;
        POSTING *newitem;
        POSTING posting;
        unsigned u;
        POSTING *newsetp, *set1p;
        long    newsetc, set1c, set2c;

        entryptr = (ENTRY *) (invcntl->logblk + 12) + invcntl->keypnt;
        ptr = invcntl->logblk + entryptr->offset;
        ptr2 = ((unsigned long *)ptr) +
            (entryptr->size + (sizeof (long) - 1)) / sizeof (long);
        *num = entryptr->post;
        switch (op) {
        switch (bool) {
        case OR:
        case NOT:
                if (*num == 0) {
                        *num = numitems;
                        return (item);
                }
        }
        /* make room for the new set */
        u = 0;
        switch (op) {
        switch (bool) {
        case AND:
        case NOT:
                newsetp = set1p = item;
                break;
```

```
        case OR:
                u = enditem - item;
                /* FALLTHROUGH */
        case REVERSENOT:
                u += *num;
                if (item == item2) {
                        if (u > setsize1) {
                                u += SETINC;
                                if ((item1 = (POSTING *) realloc(item1,
                                    u * sizeof (POSTING))) == NULL) {
                                        goto cannotalloc;
                                }
                                setsize1 = u;
                        }
                        newitem = item1;
                } else {
                        if (u > setsize2) {
                                u += SETINC;
                                if ((item2 = (POSTING *)realloc(item2,
                                    u * sizeof (POSTING))) == NULL) {
                                cannotalloc:
                                        invcannotalloc(u * sizeof (POSTING));
                                        (void) boolready();
                                        *num = -1;
                                        return (NULL);
                                }
                                setsize2 = u;
                        }
                        newitem = item2;
                }
                set1p = item;
                newsetp = newitem;
        }
        file = invcntl->postfile;
        (void) fseek(file, (long)*ptr2, SEEK_SET);
        read_next_posting(invcntl, &posting);
        newsetc = 0;
        switch (op) {
        switch (bool) {
        case OR:
                /* while something in both sets */
                set1p = item;
                newsetp = newitem;
                for (set1c = 0, set2c = 0;
                    set1c < numitems && set2c < *num; newsetc++) {
                        if (set1p->lineoffset < posting.lineoffset) {
                                *newsetp++ = *set1p++;
                                set1c++;
                        } else if (set1p->lineoffset > posting.lineoffset) {
                                *newsetp++ = posting;
                                read_next_posting(invcntl, &posting);
                                set2c++;
                        } else if (set1p->type < posting.type) {
                                *newsetp++ = *set1p++;
                                set1c++;
                        } else if (set1p->type > posting.type) {
                                *newsetp++ = posting;
                                read_next_posting(invcntl, &posting);
                                set2c++;
                        } else {        /* identical postings */
                                *newsetp++ = *set1p++;
                                set1c++;
                                read_next_posting(invcntl, &posting);
                                set2c++;
                        }
                }
```



```
                /* find out what ran out and move the rest in */
                if (set1c < numitems) {
                        newsetc += numitems - set1c;
                        while (set1c++ < numitems) {
                                *newsetp++ = *set1p++;
                        }
                } else {
                        while (set2c++ < *num) {
                                *newsetp++ = posting;
                                newsetc++;
                                read_next_posting(invcntl, &posting);
                        }
                }
                item = newitem;
                break; /* end of OR */
#if 0
        case AND:
                set1c = 0;
                set2c = 0;
                while (set1c < numitems && set2c < *num) {
                        if (set1p->lineoffset < posting.lineoffset) {
                                set1p++;
                                set1c++;
                        } else if (set1p->lineoffset > posting.lineoffset) {
                                read_next_posting(invcntl, &posting);
                                set2c++;
                        } else if (set1p->type < posting.type)  {
                                *set1p++;
                                set1c++;
                        } else if (set1p->type > posting.type) {
                                read_next_posting(invcntl, &posting);
                                set2c++;
                        } else {        /* identical postings */
                                *newsetp++ = *set1p++;
                                newsetc++;
                                set1c++;
                                read_next_posting(invcntl, &posting);
                                set2c++;
                        }
                }
                break; /* end of AND */

        case NOT:
                set1c = 0;
                set2c = 0;
                while (set1c < numitems && set2c < *num) {
                        if (set1p->lineoffset < posting.lineoffset) {
                                *newsetp++ = *set1p++;
                                newsetc++;
                                set1c++;
                        } else if (set1p->lineoffset > posting.lineoffset) {
                                read_next_posting(invcntl, &posting);
                                set2c++;
                        } else if (set1p->type < posting.type) {
                                *newsetp++ = *set1p++;
                                newsetc++;
                                set1c++;
                        } else if (set1p->type > posting.type) {
                                read_next_posting(invcntl, &posting);
                                set2c++;
                        } else {        /* identical postings */
                                set1c++;
                                set1p++;
                                read_next_posting(invcntl, &posting);
                                set2c++;
                        }
```

```
1113                }
1114                newsetc += numitems - set1c;
1115                while (set1c++ < numitems) {
1116                        *newsetp++ = *set1p++;
1117                }
1118                break; /* end of NOT */

1120        case REVERSENOT:  /* core NOT incoming set */
1121                set1c = 0;
1122                set2c = 0;
1123                while (set1c < numitems && set2c < *num) {
1124                        if (set1p->lineoffset < posting.lineoffset) {
1125                                set1p++;
1126                                set1c++;
1127                        } else if (set1p->lineoffset > posting.lineoffset) {
1128                                *newsetp++ = posting;
1129                                read_next_posting(invcntl, &posting);
1130                                set2c++;
1131                        } else if (set1p->type < posting.type) {
1132                                set1p++;
1133                                set1c++;
1134                        } else if (set1p->type > posting.type) {
1135                                *newsetp++ = posting;
1136                                read_next_posting(invcntl, &posting);
1137                                set2c++;
1138                        } else {        /* identical postings */
1139                                set1c++;
1140                                set1p++;
1141                                read_next_posting(invcntl, &posting);
1142                                set2c++;
1143                        }
1144                }
1145                while (set2c++ < *num) {
1146                        *newsetp++ = posting;
1147                        newsetc++;
1148                        read_next_posting(invcntl, &posting);
1149                }
1150                item = newitem;
1151                break; /* end of REVERSENOT  */
1152 #endif
1153        }
1154        numitems = newsetc;
1155        *num = newsetc;
1156        enditem = (POSTING *)newsetp;
1157        return ((POSTING *)item);
1158 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
    3408 Mon Aug 27 12:18:42 2012
new/usr/src/tools/cscope-fast/invlib.h
2944 cscope-fast/invlib.[ch] use reserved word as identifier
Reviewed by: Albert Lee <trisk@nexenta.com>
Reviewed by: Gary Mills <gary_mills@fastmail.fm>
Reviewed by: Joerg Schilling <Joerg.Schilling@fokus.fraunhofer.de>
**********************************************************
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License, Version 1.0 only
 * (the "License").  You may not use this file except in compliance
 * with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*      Copyright (c) 1988 AT&T */
/*        All Rights Reserved   */


/*
 * Copyright 1999, 2003 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

#ifndef INVLIB_H
#define INVLIB_H
#pragma ident   "%Z%%M% %I%     %E% SMI"

/* inverted index definitions */

/* postings temporary file long number coding into characters */
#define BASE            95      /* 127 - ' ' */
#define PRECISION       5       /* maximum digits after converting a long */

/* inverted index access parameters */
#define INVAVAIL        0
#define INVBUSY         1
#define INVALONE        2

/* boolean set operations */
#define OR              3
#define AND             4
#define NOT             5
#define REVERSENOT      6

/* note that the entire first block is for parameters */
typedef struct  {
        long    version;        /* inverted index format version */
        long    filestat;       /* file status word  */
        long    sizeblk;        /* size of logical block in bytes */
        long    startbyte;      /* first byte of superfinger */
        long    supsize;        /* size of superfinger in bytes */
```



```
        long    cntlsize;       /* size of max cntl space (should be a */
                                /* multiple of BUFSIZ) */
        long    share;          /* flag whether to use shared memory */
} PARAM;
_____unchanged_portion_omitted_

extern  long    *srcoffset;     /* source file name database offsets */
extern  int     nsrcoffset;     /* number of file name database offsets */

extern void     boolclear(void);
extern POSTING  *boolfile(INVCONTROL *invcntl, long *num, int op);
extern POSTING  *boolfile(INVCONTROL *invcntl, long *num, int bool);
extern void     invclose(INVCONTROL *invcntl);
extern long     invfind(INVCONTROL *invcntl, char *searchterm);
extern int      invforward(INVCONTROL *invcntl);
extern int      invopen(INVCONTROL *invcntl, char *invname, char *invpost,
                    int stat);
extern int      invterm(INVCONTROL *invcntl, char *term);
extern long     invmake(char *invname, char *invpost, FILE *infile);

#endif  /* INVLIB_H */
```