ภาคผนวก ค

# program listing

```
#
/* conc    Do a kwic concordance to a text file.
 * This version forks before writing to tape, thereby attempting
 * to speed up execution time by doing slow tape output in parallel with
 * filling the output buffer again.
 * Jem Clear
 * ELR, B'ham Univ
 * 20/04/83
 */

#include        <stdio.h>
#include        <ctype.h>

#define         ERR    (-1)
#define         WDLEN  16
#define         SPAN   110
#define         STOPFILE "stopf"        /* the name of a file from which a
                                        list of stop/include words will be taken */
#define         MXSTOP 60      /* max number of stop/include words */
#define         RCXT   4
#define         MTDEV"/dev/nrmt0"       /* this is system dependent!! */
                                        /* Must be non-rewind device  */
#define         MXOBKS 4100 /* for 2400ft tape */
#define         REFWDTH 12
#define         OBUFSZ 8192
#define         SP     32

char    lcase[128] = {/* quick u/case to l/case convert */

        0,1,2,3,4,5,6,7,8,9,
        10,11,12,13,14,15,16,17,18,19,
        20,21,22,23,24,25,26,27,28,29,
        30,31,32,33,34,35,36,37,38,39,
        40,41,42,43,44,45,46,47,48,49,
        50,51,52,53,54,55,56,57,58,59,
        60,61,62,63,64,'a','b','c','d','e',
         'f','g','h','i','j','k','l','m','n','o',
```

```
        'p','q','r','s','t','u','v','w','x','y',
        'z',91,92,93,94,95,96,'a','b','c',
        'd','e','f','g','h','i','j','k','l','m',
        'n','o','p','q','r','s','t','u','v','w',
        'x','y','z',123,124,125,126,127   };

char    buf[512];
int     p = 0;
int     semspan = (SPAN/2);
long    fptr = 0;
int     nread = 0;
struct  txtref { char a[6];char sp1;char b[2];char sp2;char c[2]; } rfs[127];
char    rc = 0;
char    letters[128];
char    spaces[256];/*  to put out varying length sequences of spaces */
char    *lstr = "abcdefghijklmnopqrstuvwxyz-><";
int     fold = 1;
int     xflag = 0;
int     wflag = 0;     /* explicit wordselection in stopf */
int     sbrc = 1;
char    stopwds[MXSTOP][WDLEN];
int     stops;
char    frch,toch;/* chars for selection of alpha slice */
char    *tfile1 = "/usr/tmp/wdindx1";
char    *tfile2 = "/usr/tmp/wdindx2";
char    ifile[32];
int     ifd, rfd;
char    rbuf[OBUFSZ];   /* results output buffer */
int     oc = 0;        /* count of characters in output buffer */
int     obc = 0;       /* output block count */
FILE    *xf[2];
struct  index { char wd[WDLEN]; char rcxt[RCXT+1]; long loc; char ref; };
struct  index rec, orec;

main(argc,argv)
int argc;
char **argv;
```

```
{
        int isstop;
        register struct index *rp, *orp;
char *sortcom1 = "sort -T /usr3/tmp -o /usr/tmp/wdindx1 /usr/tmp/wdindx1";
char *sortcom2 = "sort -T /usr3/tmp -o /usr/tmp/wdindx2 /usr/tmp/wdindx2";
        rp = &rec;
        orp = &orec;
        rfd=1;
        while (--argc > 0 && (*++argv)[0]=='-') {
                char *s;
                for (s=argv[0]+1;*s;s++)
                        switch (*s) {

                        case 't':
                                if ((rfd=open(MTDEV,1))==ERR)
                                        die("can't open",MTDEV);
                                continue;
                        case 'x':
                                xflag++;
                                continue;
                        case 'w':
                                wflag++;
                                continue;
                        default:
                                fprintf(stderr,"unknown option %c\n",*s);
                                continue;
                        }
        }
        if (argc > 0) {
                strcpy(ifile,*argv);
                if ((ifd=open(ifile,0))==ERR) die("can't open",ifile);
        }
        else die("text file argument missing","");
        if ((xf[0]=fopen(tfile1,"w"))==NULL) die("can't create",tfile1);
        if ((xf[1]=fopen(tfile2,"w"))==NULL) die("can't create",tfile2);
        setlett();
        getstop();
```

```
        strcpy(rp->rcxt,"*");
        strcpy(orp->rcxt,"*");
        orp->wd[0] = '\0';/* to indicate first-time-thru' */
        for (;;) {
                if (rp == &rec) { rp = &orec; orp = &rec; }
                else { rp = &rec; orp = &orec; }
                if (grab(rp)==0) break;
                isstop = instop(orp);
                if ((!isstop&&!wflag)||(isstop&&wflag)) {
                        if (sbrc && orp->wd[0]) getrcxt(rp->wd,orp->rcxt);
                        if (orp->wd[0]) sift(orp);
                }
        }
        fclose(xf[0]);
        fclose(xf[1]);
        if (system(sortcom1) || system(sortcom2))
                die("error in sorting","");
        chkrefs();
        if (!xflag) doconc();
}

grab(recp)
struct index *recp;
{
        register char *w;
        register int i;
        int ref=0;
        char *endwd = recp->wd+(WDLEN-1);

        w = recp->wd;
        for (i=0;i<WDLEN;i++) *w++ = '\0';
        w = recp->wd;
cont:   for (;p<nread;p++) {
                if (ref) {
                        if (w==endwd || buf[p]=='>') {
                                *w = '>';
                                dorefs(recp->wd);
```

```
                        w = recp->wd;
                        for (i=0;i<WDLEN;i++) *w++ = '\0';
                        w = recp->wd;
                        ref=0;
                }
                else {
                        *w++ = buf[p];
                }
                continue;
        }
        if (letters[buf[p]]) {
                if (buf[p]=='<') { ref=1; recp->wd[0]='<'; }
                if (w==recp->wd) {
                        recp->loc = (fptr+p)-nread;
                        recp->rcf = rcl128;
                }
                if (buf[p]==047) continue;
                if (w<endwd) *w++ = lcase[buf[p]];
                continue;
        }
        if (w > recp->wd) return(1);
        else continue;
    }
    if ((nread=read(ifd,buf,512)) <= 0) return(0);
    p = 0;
    fptr += nread;
    goto cont;
}


die(s,t)
char *s, *t;
{
    fprintf(stderr,"%s %s\n",s, t);
    exit();
}
```

```
doconc()
{
        int n;
        register char *p;
        register int scst;
        register int span;
        extern int semspan;
        long pos;

        xf[1]=NULL;
        if ((xf[0]=fopen(tfile1,"r"))==NULL) die("can't reopen",tfile1);
next:   while ((scst=fscanf(xf[0],"%*s %*s %ld %c\n",&(rec.loc),&(rec.ref)))==2)
        {
                outp(rfd,&rfs[rec.ref&127],12);
                pos=rec.loc-semspan;;
                if (pos<0) {                    /* leading spaces? */
                        outp(rfd,spaces,(int)(pos-(pos*2)+1));
                        span = (SPAN)+pos;
                        pos=0;
                }
                else {
                        span = (SPAN);
                        outp(rfd,spaces,1);
                }
                if (lseek(ifd,pos,0)==ERR) die("bad seek in doconc","");
                if ((n=read(ifd,buf,span))==ERR) die("bad read","");
                for (p=buf;p<(buf+n);p++) {
                        if (*p != '\n') outp(rfd,p,1);
                        else outp(rfd," ",1);
                }
                outp(rfd,"\n",1);
        }
        if (scst!=EOF) {
                fprintf(stderr,"conc: duff index\n");
                goto next;
        }
```

```
        if (xf[1]==NULL) {
                fclose(xf[0]);
                if ((xf[0]=fopen(tfile2,"r"))==NULL)
                        die("can't reopen",tfile2);
                xf[1]=xf[0];
                goto next;
        }
        outp(rfd,"\n",1);
        flush(rfd);
        wait(0);
        close(rfd);
        fclose(xf[1]);
}


instop(recp)
struct index *recp;
{
        register int i, val;

        if (!wflag && recp->wd[0]=='<') return(1);
        if (toch && (recp->wd[0]<frch || recp->wd[0]>toch)) return(1);
        for (i=0;i<stops;i++) {
                if (val=strcmp(recp->wd,stopwds[i])) {
                        if (val<0) return(0); else continue;
                } else return(1);
        }
        return(0);
}


getstop()
{
        FILE *tmpf;
        int compar();
        register int i;
        if ((tmpf=fopen(STOPFILE,"r"))==NULL) {
```

```
                fprintf(stderr,"conc:(message) default stop words\n");
                strcpy(stopwds[0],"'");
                strcpy(stopwds[1],"-");
                strcpy(stopwds[2],"a");
                strcpy(stopwds[3],"of");
                strcpy(stopwds[4],"the");
                stops=5;
                return;
        }
        if (fscanf(tmpf,"*%c-%c\n",&frch,&toch)==2)
                fprintf(stderr,"concordance select from %c to %c\n",frch,toch);
        else {
                fseek(tmpf,0L,0);
                toch = frch = '\0';
        }
        while (fscanf(tmpf,"%s",stopwds[stops++])==1 && stops<MXSTOP) {
                for (i=0;i<WDLEN;i++)
                        stopwds[stops][i] = lcase[stopwds[stops][i]];
                stops++;
        }
        fclose(tmpf);
        qsort(stopwds,stops,WDLEN,compar);
}


setlett()
{
        register char *p;
        register int i;

        for (i=0;i<128;i++) letters[i]=0;
        p=lstr;
        while (*p) {
                letters[*p] = 1;
                if (fold && isalpha(*p)) letters[(*p)-32] = 1;
                p++;
        }
```

```
        for (i=0;i<256;i++) spaces[i]=' ';
}

dorefs(str)
char *str;
{
        if (rc>=127) return;
        switch (str[1]) {

        case 't':
        case 'T':
                if (sscanf(str,"<%*c %6s>",rfs[++rc].a) != 1) {
                        break;
                }
                rfs[rc].sp1 = SP;
                return;
        case 'O':
        case 'o':
                if (sscanf(str,"<%*c %2s>",rfs[rc].b)!=1) {
                        break;
                }
                rfs[rc].sp2 = SP;
                return;
        case 'x':
        case 'X':
                if (sscanf(str,"<%*c %2s>",rfs[rc].c)!=1)
                        break;
        default:
                return;
        }
        fprintf(stderr,"bad reference %s\n",str);
}


outp(fd,addr,n)
int fd;
char *addr;
```

```
register int n;
{
        extern int oc;
        register char *p = addr;

cont:   for (;oc<OBUFSZ;oc++) {
                if (n--) rbuf[oc] = *p++;
                else return;
        }
        flush(fd);
        oc = 0;
        goto cont;
}


flush(fd)
int fd;
{
        if (oc%2) outp(fd,"\n",1);
        wait(0);
        if (fork()) return;
        if (write(fd,rbuf,oc)==ERR) {
                fprintf(stderr,"conc: tape write error\n");
                close(fd);
                newreel();
                exit(0);
        }
        if (obc++ <= MXOBKS || fd==1) exit(0);
        close(fd);
        newreel();
        exit(0);
}


getrcxt(fr,to)
char fr[], to[];
{
```

```
        register int i;

        for (i=0;i<RCXT;i++) {
                if (fr[i]=='\n') to[i] = SP;
                else to[i] = lcase[fr[i]];
                if (to[i]=='\0') break;
        }
}


sift(recp)
struct index *recp;
{
        char c = recp->wd[0];

        c = lcase[c];
        if (c < 'n')
                fprintf(xf[0],"%s %s %ld %c\n",recp->wd,recp->rcxt,recp->loc,recp->ref);
        else
                fprintf(xf[1],"%s %s %ld %c\n",recp->wd,recp->rcxt,recp->loc,recp->ref);
}

newreel()
{
        char c;

        do {
                fprintf(stderr,"Tape full. Load new tape and hit return ");
                while ((c=getchar())!='\n') fputc('\n',stderr);
        } while ((rfd=open(MTDEV,1))==ERR);
        obc = 0;
        return;
}


chkrefs()
{
```

```
        register int i, j;
        register char *p;

        for (i=1;i<=rc;i++) {
                p = &rfs[i];
                for (j=0;j<REFWDTH;j++) {
                        if (isprint(*p)==0) *p = SP;
                        p++;
                }
        }
}

compar(s1,s2)
char *s1, *s2;
{
        return(strcmp(s1,s2));
}
```

```
#
/*  mk_indx.c  -  to create a higher level index from the wdindx* files
*  produced by 'conc'. The headword is placed in alpha sequence in the
*  hi_index file with a pointer to the lo_index file and a freq count.
*  The lo_indx file contains only the byte-offset into the source text
*  plus the right context and ref char.  These two index files are then
*  used with the pr_conc program to display concordances interactively.
*  Jem Clear
*  ELR, B'ham Univ
*  22/04/86
*/

#include        <stdio.h>
#include"       conc.h"         /* put this somewhere else perhaps? */

FILE            *inx;
int             hlfd, llfd;

main(argc,argv)
int argc; char **argv;
{
                struct lindex low;
                struct hindex hi;
                char word[WDLEN], lastwd[WDLEN];
                int both = 0;
                int n;
                long k = 0L;

                argv++;
                if ((inx=fopen(tfile1,"r"))==NULL)
                        die("Can't open",tfile1);
                if ((hlfd=creat(hlevf,0644))==ERR || (llfd=creat(llevf,0644))==ERR)
                        die("Can't creat index files","");

                lastwd[0]='\0';
cont:while ((n=fscanf(inx,"%s %s %ld %c\n",word,low.rc,&low.tp,&low.rf))==4)
                {
                        if (strcmp(word,lastwd)==0) {  /* if they're identical */
                                hi.c++;
                        }
                        else {
                                if (lastwd[0]) {
                                        strcpy(hi.w,lastwd);
                                        if (write(hlfd,&hi,HILEN)==ERR)
                                                die("write error on",hlevf);
                                }
                                hi.ptr = k;
                                hi.c = 1;
                                strcpy(lastwd,word);
                }
                if (write(llfd,&low,LOLEN)==ERR)
                        die("write error on",llevf);
                k++;
        }
        if (n!=EOF) {
```

```
                fprintf(stdout,"duff index! %s\n",word);
                goto cont;
        }
        if (both) {
                strcpy(hi.w,lastwd);
                if (write(hlfd,&hi,HILEN)==ERR)
                        die("write error on",hlevf);
                exit(0);
        }
        if (freopen(tfile2,"r",inx)==NULL)
                die("Can't open",tfile2);
        both++;
        goto cont;
}


die(s,t)
char *s, *t;
{
        fprintf(stderr,"%s %s\n",s, t);
        exit();
}
```

```
#
 /*  wdl       do a word frequeny count on a text file.
*  This calls the standard Unix sort command to do the work. I supply my
*  own uniq routine.
*  Jem Clear
*  ELR, B'ham Univ
*  24/04/86
*/

#include    <stdio.h>
#include    <ctype.h>
#include    <signal.h>

#define     ERR         (-1)
#define     WDLEN       16
#define     CHUNK       1000

char        letters[128];
char        *lstr = "abcdefghijklmnopqrstuvwxyz-'><";
char        commstr[128];
unsigned rflag, fflag, oflag, fold, sflag;
char        indxf[48], tempf[48];
int         ifd = 0;

main(argc,argv)
int argc;
char **argv;
{
            char ifile[32], *s;
            char word[WDLEN];
            long ord;
            FILE *inx;
            extern trap();
            char *getcomm();

            fold++;
            while (--argc > 0 && (*++argv)[0]=='-')
                for (s=argv[0]+1;*s;s++)
                        switch (*s) {

                        case 'a':
                                continue;
                        case 'f':
                                fflag++;
                                continue;
                        case 'r':
                                rflag++;
                                continue;
                        case 'o':
                                oflag++;
                                continue;
                        case 's':
                                sflag++;
                                oflag++;
                                continue;
```

```
                default:
                        fprintf(stderr,"unknown option %c\n",*s);
                        continue;
        }
        if (argc) {
        strcpy(ifile,*argv);
        if ((ifd=open(ifile,0))==ERR) die("wl: can't open",ifile);
        }
        else ifd=0;
        getnams(indxf,tempf);
        if ((inx=fopen(indxf,"w"))==NULL) die("can't open",indxf);
        signal(2,trap);
        signal(3,trap);

        setlett(lstr);
        for (;;) {
            if (grab(word)==0) break;
            if (word[0]=='<') continue;
            ord++;
            if (oflag) fprintf(inx,"%s %ld",word,ord);
            else fputs(word,inx);
            fputs("\n",inx);
        }
        fclose(inx);
        system(getcomm(commstr));
        if (oflag) xuniq(indxf);
        else uniq(indxf);
        if (fflag || oflag) fsort();
        unlink(indxf);
        if (fflag || oflag) unlink(tempf);
}


grab(wd)
char *wd;
{
        static char buf[512];
        register char *w;
        static int p, nread;
        register int i;
        int ref=0;
        char *endwd = wd+(WDLEN-1);

        w = wd;
        for (i=0;i<WDLEN;i++) *w++ = '\0';
        w = wd;
cont:   for (;p<nread;p++) {
            if (letters[buf[p]]) {
                    if (w<endwd) *w++ = buf[p];
                    continue;
            }
            if (w > wd) return(1);
            else continue;
            }
```

```
        if ((nread=read(ifd,buf,512)) <= 0) return(0);
        p = 0;
        goto cont;
}


ident(s,t)
register char *s, *t;
{
        register int n = 0;

        while (*s)
            if (*t) { s++; t++; n++; }
            else return(0);

        if (*t) return(0);
        n++;
        for (;n;n--,t--,s--) {
            switch (*t-*s) {

            case -32:
                    if (*s>='a' && *s<='z') continue;
                    return(0);
            case 0:
                    continue;
            case 32:
                    if (*t>='a' && *t<='z') continue;
            default:
                    return(0);
            }
        }
        return(1);
}



uniq(indxf)
char *indxf;
{
        char word[WDLEN+2], lastwd[WDLEN+2];
        long types = 0;
        long tokens = 0;
        long c = 0;
        FILE *inx, *tmpf;
        char *input();

        if ((inx=fopen(indxf,"r"))==NULL) die("can't open",indxf);
        if (fflag) {
            if ((tmpf=fopen(tempf,"w"))==NULL) die("can't open",tempf);
        }
        else
            tmpf = stdout;

        lastwd[0] = '\0';
        while (input(word,WDLEN+2,inx)) {
```

```
                tokens++;
                if (ident(word,lastwd)) {
                        c++;
                        continue;
                }
                if (lastwd[0]) {
                        types++;
                        if (fflag)
                                fprintf(tmpf,"%ld %s\n",c,lastwd);
                        else
                                fprintf(tmpf,"%s %ld\n",lastwd,c);
                }
                strcpy(lastwd,word);
                c = 1;
        }
        if (fflag)
                fprintf(tmpf,"%ld %s\n",c,lastwd);
        else
                fprintf(tmpf,"%s %ld\n",lastwd,c);
        types++;
        fprintf(stderr,"%ld tokens\n%ld types\n",tokens,types);
        fclose(inx);
        if (tmpf!=stdout) fclose(tmpf);
}


die(s,t)
char *s, *t;
{
        fprintf(stderr,"wl: %s %s\n",s, t);
        exit();
}


setlett(lstr)
char *lstr;
{
        register char *p;
        register int i;

        for (i=0;i<128;i++) letters[i]=0;
        p=lstr;
        while (*p) {
                letters[*p] = 1;
                if (fold && isalpha(*p)) letters[(*p)-32] = 1;
                p++;
        }
}


fsort()
{
        char buf[512];
        register int nread, ty;
```

```
        int fd, f;
        FILE *fp;
        long n, focc;
        ty = f = nread = 0;
        n = CHUNK;
        if (rflag) {
            sprintf(commstr,"sort -nfr -o %s -T /usr/tmp %s",tempf,tempf);
        }
        else sprintf(commstr,"sort -nf -o %s -T /usr/tmp %s",tempf,tempf);
        system(commstr);
        if (sflag) {
            if ((fp=fopen(tempf,"r"))==NULL) die("can't open",tempf);
            while (fscanf(fp,"%ld %s %ld",&focc,buf,&f)==3)
                    if (focc >= n) {
                            printf("%ld,%d\n",n,ty);
                            ty = 1;
                            n += CHUNK;
                    }
                    else ty++;
        }
        else {
            if ((fd=open(tempf,0))==ERR) die("can't open",tempf);
            while ((nread=read(fd,buf,512))>0)
                    write(1,buf,nread);
        }
        return;
}



getnams(ind,tmp)
char *ind, *tmp;
{
        char *pid, *ecvt();
        int x;

        strcpy(ind,"/usr/tmp/");
        pid = ecvt((double)getpid(),5,&x,&x);
        strcat(ind,"wl");
        strcat(ind,pid);
        strcpy(tmp,ind);
        strcat(ind,"i");
        strcat(tmp,"t");
}

trap()
{
        putchar('\n');
        unlink(indxf);
        if (fflag) unlink(tempf);
        exit(-1);
}

char *getcomm(s)
char *s;
{
```

```
        if (!rflag)
            sprintf(s,"sort +0 -1f +1n -o %s -T /usr/tmp %s",indxf,indxf);
        else
            sprintf(s,"sort +0 -1rf +1n -o %s -T /usr/tmp %s",indxf,indxf);
        return(s);
}



char *input(s,n,fp)
register char *s;
register int n;
FILE *fp;
{
        register int gotsum=0;
        for (;n-1;n--) {
            if ((*s=getc(fp))=='\n') break;
            if (*s==EOF) {
                    if (gotsum) break;
                    else return(NULL);
            }
            gotsum++;
            s++;
        }
        *s='\0';
        return(s);
}



xuniq(indxf)
char *indxf;
{
        char word[WDLEN+2], lastwd[WDLEN+2];
        long seq, focc;
        long types = 0;
        long tokens = 0;
        long c = 0;
        FILE *inx, *tmpf;

        if ((inx=fopen(indxf,"r"))==NULL) die("can't open",indxf);
        if ((tmpf=fopen(tempf,"w"))==NULL) die("can't open",tempf);

        lastwd[0] = '\0';
        while (fscanf(inx,"%s %ld\n",word,&seq)==2) {
            tokens++;
            if (idcnt(word,lastwd)) {
                    c++;
                    continue;
        }
        if (lastwd[0]) {
            types++;
            rintf(tmpf,"%ld %s %ld\n",focc,lastwd,c);

        }
```

```
#
/*  pr_conc   to retrieve concordances interactively from the index files
*  created by mk_indx. The files hi_indx_ and lo_indx_ should exist
*  already (pr_conc will simply abort if it doesn't find them). Use the
*  conc.h file to define hlevf and llevf filenames.
*  Jem Clear
*  ELR, B'ham Univ
*  24/04/86
*/

#include    <stdio.h>
#include    "conc.h"

#define     OBUFSZ    2048    /* any size will do! */
#define     LCXT      4       /* same as RCXT in conc.h */

int         hlfd, llfd, txfd;
int         oc;
char        rbuf[OBUFSZ];
char        buf[512];
int         span = 100;
char        textf[128];
char        *sortcmd = "sort -df -o /usr/tmp/pr_conc.t /usr/tmp/pr_conc.t";
char        *tmpf = "/usr/tmp/pr_conc.t";  /* this gash file used to sort */
                                           /* on left context if required */
                                           /* If you change it, you must also */
                                           /* fix sortcom initialization above */
FILE        *tpfp;
char        reply[24];
int         left = 0;
extern long lseek();

main(argc,argv)
int argc;
char **argv;
{
            char target[48];
            char r[8];
            int ncit = 0;
            register int i, j, n;
            struct hindex req;

            if (--argc==1) {
                argv++;
                strcpy(textf,*argv);
            }
            else die("missing text argument","");
            if ((hlfd=open(hlevf,0))==ERR)
                die("Can't open",hlevf);
            if ((txfd=open(textf,0))==ERR) die("can't open",textf);
            if ((llfd=open(llevf,0))==ERR) die("can't open",llevf);
            for (;;) {
                fputs("Word? ",stderr);
                if (gets(target)==NULL) break;
                if (strlen(target) > WDLEN) {
```

```
                fputs("too long\n",stderr);
                continue;
            }
            if (find(target,&req)==0) {
                fprintf(stderr,"can't find %s\n",target);
                continue;
            }
            fputs("Sorted left or right (l/r)? ",stderr);
            if (gets(reply)==NULL) break;
            if (*reply=='l') left++;
            else left = 0;
        fprintf(stderr,"%s has %ld citations - how many required ",target,req.c);
            do {
                fputs("? ",stderr);
                gets(r);
                ncit=atoi(r);
                if (ncit > req.c) ncit=0;
            } while (ncit<=0);
            fputs("context span? ",stderr);
            gets(r);
            span = atoi(r);
            doconc(req.ptr,req.c,ncit);
        }
        fputc('\n',stderr);
}


find(s,rec)
char *s;
struct hindex *rec;
{
        long top, bot, mid;

        bot = 0;
        top = (lseek(hlfd,0L,2)/HILEN)-1;

        while (bot <= top) {
            mid = (top+bot)/2;
            if (getent(mid,rec)==0) return(0);
            switch (compare(rec->w,s)) {

            case 0:
                return(1);
            case 1:
                top = mid-1;
                break;
            case -1:
                bot = mid+1;
                break;
            }
        }
        return(0);
}
```

```
getent(n,iptr)
long n;
struct hindex *iptr;
{
        if (lseek(hlfd,n*HILEN,0)==ERR) die("bad seek in getent","");
        if (read(hlfd,iptr,HILEN)!=HILEN)
            die("bad read in getent","");
        return(1);
}


doconc(loc,c,req)
long loc, c;
int req;
{
        int n;
        register char *p;
        register int i, j;
        long pos;
        struct lindex low;
        int everyn = (int)(c/req);
        int rfd=1;

        if (left) if ((tpfp=fopen(tmpf,"w"))==NULL) die("can't open",tmpf);
        if (lseek(llfd,loc*LOLEN,0)==ERR) die("bad seek:",llevf);
        for (i=0;i<c;i++) {
            if (read(llfd,&low,LOLEN)!=LOLEN)
                    die("bad read in doconc","");
            if (i%everyn) continue;
            if (left) {
                    if ((pos=low.tp-LCXT)<0) pos=0L;
                    if (lseek(txfd,pos,0)==ERR) die("bad seek","");
                    for (j=0;j<LCXT;j++) {
                            read(txfd,&buf[j],1);
                            if (buf[j]=='\n' || buf[j]=='\r') buf[j]=' ';
                    }
                    buf[j]='\0';
                    rev(buf);
                    fprintf(tpfp,"%s\%ld\n",buf,low.tp);
                    continue;
            }
            pos=low.tp-(span/2);
            if (pos < 0) pos=0;
            if (lseek(txfd,pos,0)==ERR) die("bad seek in doconc","");
            if ((n=read(txfd,buf,span))==ERR) die("bad read","");
            outp(rfd," ",1);
            for (p=buf;p<(buf+n);p++) {
                    if (*p!='\n'&&*p!='\r') outp(rfd,p,1);
                    else outp(rfd," ",1);
            }
            outp(rfd,"\n",1);
        }
        if (left) {
            fclose(tpfp);
            system(sortcmd);
```

```
                leftpr(txfd,rfd);
        }
        outp(rfd,"\n",1);
        flush(rfd);
        oc = 0;
}


compare(s1,s2)
char *s1, *s2;
{
        int n;
        if ((n=strcmp(s1,s2))<0) return(-1);
        if (n>0) return(1);
        return(0);
}


die(s,t)
char *s, *t;
{
        fprintf(stderr,"pr_conc: %s %s\n",s,t);
        exit(-1);
}


outp(fd,addr,n)
int fd;
char *addr;
register int n;
{
        extern int oc;
        register char *p = addr;

cont:for (;oc<OBUFSZ;oc++) {
                if (n--) rbuf[oc] = *p++;
                else return;
        }
        flush(fd);
        oc = 0;
        goto cont;
}


flush(fd)
int fd;
{
        if (oc%2) { outp(fd,"\n",1); oc++; }
        if (write(fd,rbuf,oc)==ERR) {
                fprintf(stderr,"wx: write error\n");
                close(fd);
                exit(0);
        }
        return;
}
```

```
leftpr(txfd,rfd)
int txfd;
int rfd;
{
        FILE *tpfp;
        long pos;
        register int n;
        register char *p;
        int stat;

        if ((tpfp=fopen(tmpf,"r"))==NULL) die("can't reopen",tmpf);

        while ((stat=fscanf(tpfp,"%[^\l]|%ld\n",buf,&pos))==2) {
            pos=pos-(span/2);
            if (pos < 0) pos=0;
            if (lseek(txfd,pos,0)==ERR) die("bad seek in leftpr","");
            if ((n=read(txfd,buf,span))==ERR) die("bad read in leftpr","");
            outp(rfd," ",1);
            for (p=buf;p<(buf+n);p++) {
                    if (*p!='\n'&&*p!='\r') outp(rfd,p,1);
                    else outp(rfd," ",1);
            }
            outp(rfd,"\n",1);
        }
        if (stat!=EOF) die("bad fscanf in leftpr","");
        fclose(tpfp);
        unlink(tmpf);
}


rev(s)
char s[];
{
        register char c;
        register char *p, *q;

        for (q=s;*q;q++);
        p=s; q--;
        while (p<q) {
            c=(*q);
            (*q--)=(*p);
            *p++=c;
        }
}
```

```
/* conc.h    An include file for the concordance progs.
* Jem Clear
* ELR, B'ham Univ
* 24/04/86
*/

#define     ERR         (-1)
#define     WDLEN       16
#define     RCXT        4
#define     HILEN       24
#define     LOLEN       13

char    buf[512];
struct  txtref { char a[6];char sp1;char b[2];char sp2;char c[2]; } rfs[127];
char    rc = 0;
char    letters[128];
char    *lstr = "abcdefghijklmnopqrstuvwxyz-'><";
char    *tfile1 = "/usr/tmp/wdindx1";
char    *tfile2 = "/usr/tmp/wdindx2";
char    *hlevf  = "hi_indx_";
char    *llevf  = "lo_indx_";
struct  index { char wd[WDLEN]; char rcxt[RCXT+1]; long loc; char ref; };
struct  lindex { char rc[RCXT+1]; long tp; char rf; };
struct  hindex { char w[WDLEN];  long ptr; long c; };
```

```
#include <stdio.h>
main(argc,argv)
int argc;
char *argv[];
{
        int nc, nl;
        char c;
        FILE *fp;
        if (argc != 2){
            printf("Error argument \n");
            exit(-1);
        }
        if ((fp = fopen(argv[--argc],"r"))==NULL){
            printf("Can not open file %s\n",argv[argc]);
            exit(-1);
        }
        nc = 0;
        nl = 1;
        while ((c = getc(fp)) != EOF){
            if (nc >= 132){
                    putc('\n',stdout);
                    if (nl >= 60){
                            putc(",stdout);
                            nl = 0;
                    }
                    ++nl;
                    nc = 0;
                    }
        putchar(c);
        ++nc;
        if (c == '\n'){
                    nc = 0;
                    ++nl;
            }
        }
   putc('\n',stdout);
}
```

```
awk '{if ($2 > 10)
        n = 10;
    else
        n = $2;
    print $1;
    print "r";
    print n;
    print 320;
}' $1
```

```
#
/* pr_concn      to retrieve concordances interactively from the index files
* created by mk_indx. The files hi_indx_ and lo_indx_ should exist
* already (pr_conc will simply abort if it doesn't find them). Use the
* conc.h file to define hlevf and llevf filenames.
* Jem Clear
* ELR, B'ham Univ
* 24/04/86
*/

#include    <stdio.h>
#include    "conc.h"

#define     OBUFSZ    2048    /* any size will do! */
#define     LCXT      4       /* same as RCXT in conc.h */

int         hlfd, llfd, txfd;
int         oc;
char        rbuf[OBUFSZ];
char        buf[512];
int         span = 100;
char        textf[128];
char        *sortcmd = "sort -df -o /usr/tmp/pr_conc.t /usr/tmp/pr_conc.t";
char        *tmpf = "/usr/tmp/pr_conc.t";          /* this gash file used to sort */
                                                   /* on left context if required */
                                                   /* If you change it, you must also */
                                                   /* fix sortcom initialization above */
FILE        *tpfp;
char        reply[24];
int         left = 0;
extern long lseek();
char target[48];
char  *numreq;

main(argc,argv)
int argc;
char **argv;
{
            char r[8];
            int ncit = 0;
            register int i, j, n;
            struct hindex req;

            if (--argc==1) {
                argv++;
                strcpy(textf,*argv);
            }
            else die("missing text argument","");
            if ((hlfd=open(hlevf,0))==ERR)
                die("Can't open",hlevf);
            if ((txfd=open(textf,0))==ERR) die("can't open",textf);
            if ((llfd=open(llevf,0))==ERR) die("can't open",llevf);
            for (;;) {
                fputs("Word? ",stderr);
                if (gets(target)==NULL) break;
                if (strlen(target) > WDLEN) {
```

```
                        fputs("too long\n",stderr);
                        continue;
                }
                if (find(target,&req)==0) {
                        fprintf(stderr,"can't find %s\n",target);
                        continue;
                }
                fputs("Sorted left or right (l/r)? ",stderr);
                if (gets(reply)==NULL) break;
                if (*reply=='l') left++;
                else left = 0;
fprintf(stderr,"%s has %ld citations - how many required ",target,req.c);
                do {
                        fputs("? ",stderr);
                        gets(r);
                        ncit=atoi(r);
                        strcpy(r,numreq);
                        if (ncit > req.c) ncit=0;
                } while (ncit<=0);
                fputs("context span? ",stderr);
                gets(r);
                span = atoi(r);
                doconc(req.ptr,req.c,ncit);
        }
        fputc('\n',stderr);
}


find(s,rec)
char *s;
struct hindex *rec;
{
        long top, bot, mid;

        bot = 0;
        top = (lseek(hlfd,0L,2)/HILEN)-1;

        while (bot <= top) {
            mid = (top+bot)/2;
            if (getent(mid,rec)==0) return(0);
            switch (compare(rec->w,s)) {

            case 0:
                    return(1);
            case 1:
                    top = mid-1;
                    break;
            case -1:
                    bot = mid+1;
                    break;
            }
        }
        return(0);
}
```

```
getent(n,iptr)
long n;
struct hindex *iptr;
{
        if (lseek(hlfd,n*HILEN,0)==ERR) die("bad seek in getent","");
        if (read(hlfd,iptr,HILEN)!=HILEN)
            die("bad read in getent","");
        return(1);
}

doconc(loc,c,req)
long loc, c;
int req;
{
        int n;
        register char *p;
        register int i, j;
        long pos;
        struct lindex low;
        int everyn = (int)(c/req);
        int rfd=1;

        if (left) if ((tpfp=fopen(tmpf,"w"))==NULL) die("can't open",tmpf);
        if (lseek(llfd,loc*LOLEN,0)==ERR) die("bad seek:",llevf);
        if (write(rfd,"***    ",7)==ERR) {
            fprintf(stderr,"wx: write error\n");
            close(rfd);
            exit(0);
        }
        if (write(rfd,target,strlen(target))==ERR) {
            fprintf(stderr,"wx: write error\n");
            close(rfd);
            exit(0);
        }
        if (write(rfd,"   *** has  ",15)==ERR) {
            fprintf(stderr,"wx: write error\n");
            close(rfd);
            exit(0);
        }
        if (write(rfd,numreq,strlen(numreq))==ERR) {
            fprintf(stderr,"wx: write error\n");
            close(rfd);
            exit(0);
        }
        if (write(rfd,"\n",1)==ERR) {
            fprintf(stderr,"wx: write error\n");
            close(rfd);
            exit(0);
        }
        for (i=0;i<c;i++) {
            if (read(llfd,&low,LOLEN)!=LOLEN)
                    die("bad read in doconc","");
            if (i%everyn) continue;
            if (left) {
                    if ((pos=low.tp-LCXT)<0) pos=0L;
```

```
                        if (lseek(txfd,pos,0)==ERR) die("bad seek","");
                        for (j=0;j<LCXT;j++) {
                                read(txfd,&buf[j],1);
                                if (buf[j]=='\n' || buf[j]=='\r') buf[j]=' ';
                        }
                        buf[j]='\0';
                        rev(buf);
                        fprintf(tpfp,"%s|%ld\n",buf,low.tp);
                        continue;
                }
                        pos=low.tp-(span/2);
                        if (pos < 0) pos=0;
                        if (lseek(txfd,pos,0)==ERR) die("bad seek in doconc","");
                        if ((n=read(txfd,buf,span))==ERR) die("bad read","");
                        outp(rfd," ",1);
                        for (p=buf;p<(buf+n);p++) {
                                if (*p!='\n'&&*p!='\r') outp(rfd,p,1);
                                else outp(rfd," ",1);
                        }
                        outp(rfd,"\n",1);
                        outp(rfd,"\n",1);
                }
                if (left) {
                        fclose(tpfp);
                        system(sortcmd);
                        leftpr(txfd,rfd);
                }
                outp(rfd,"\n",1);
                flush(rfd);
                oc = 0;
}


compare(s1,s2)
char *s1, *s2;
{
                int n;
                if ((n=strcmp(s1,s2))<0) return(-1);
                if (n>0) return(1);
                return(0);
}


die(s,t)
char *s, *t;
{
                fprintf(stderr,"pr_conc: %s %s\n",s,t);
                exit(-1);
}

outp(fd,addr,n)
int fd;
char *addr;
register int n;
{
```

```
        extern int oc;
        register char *p = addr;

cont:for (;oc<OBUFSZ;oc++) {
            if (n--) rbuf[oc] = *p++;
            else return;
        }
        flush(fd);
        oc = 0;
        goto cont;
}


flush(fd)
int fd;
{
        if (oc%2) { outp(fd,"\n",1); oc++; }
        if (write(fd,rbuf,oc)==ERR) {
            fprintf(stderr,"wx: write error\n");
            close(fd);
            exit(0);
        }
        return;
}


leftpr(txfd,rfd)
int txfd;
int rfd;
{
        FILE *tpfp;
        long pos;
        register int n;
        register char *p;
        int stat;

        if ((tpfp=fopen(tmpf,"r"))==NULL) die("can't reopen",tmpf);

        while ((stat=fscanf(tpfp,"%[^\]|%ld\n",buf,&pos))==2) {
            pos=pos-(span/2);
            if (pos < 0) pos=0;
            if (lseek(txfd,pos,0)==ERR) die("bad seek in leftpr","");
            if ((n=read(txfd,buf,span))==ERR) die("bad read in leftpr","");
            outp(rfd," ",1);
            for (p=buf;p<(buf+n);p++) {
                    if (*p!='\n'&&*p!='\r') outp(rfd,p,1);
                    else outp(rfd," ",1);
            }
            outp(rfd,"\n",1);
        }
        if (stat!=EOF) die("bad fscanf in leftpr","");
        fclose(tpfp);
        unlink(tmpf);
}
```

```
rev(s)
char s[];
{
        register char c;
        register char *p, *q;

        for (q=s;*q;q++);
        p=s; q--;
        while (p<q) {
            c=(*q);
            (*q--)=(*p);
            *p++=c;
        }
}
```

```
#
cd
cd new-agro
pr_concn ../agro-ind/agro-total >> tmp_new



#
cd
cd new-agro
pr_concn ../agro-ind/agro-total
```

```
31a32,33
> char target[48];
> char    *numreq;
37d38
< char target[48];
67c68
<     fprintf(stderr,"%s has %ld citations - how many required ",target,req.c);

>         fprintf(stderr,"%s has %ld citations - how many required ",target,req.c);
71a73
>  strcpy(r,numreq);
137a140,164
>         if (write(rfd,"***    ",7)==ERR) {
>             fprintf(stderr,"wx: write error\n");
>             close(rfd);
>             exit(0);
>         }
>         if (write(rfd,target,strlen(target))==ERR) {
>             fprintf(stderr,"wx: write error\n");
>             close(rfd);
>              exit(0);
>         }
>         if (write(rfd,"   *** has  ",15)==ERR) {
>             fprintf(stderr,"wx: write error\n");
>             close(rfd);
>             exit(0);
>         }
>         if (write(rfd,numreq,strlen(numreq))==ERR) {
>             fprintf(stderr,"wx: write error\n");
>             close(rfd);
>              exit(0);
>         }
>         if (write(rfd,"\n",1)==ERR) {
>             fprintf(stderr,"wx: write error\n");
>             close(rfd);
>             exit(0);
>         }
162a190
>             outp(rfd,"\n",1);
```