/035

Bell Laboratories

subject: Description of the LPR Program

date: July 23, 1974

from: I. A. Winheim

PROGRAMMERS NOTES

I. Introduction

The purpose of this document is to describe lpr.c as it exists in UNIX Release 2, as well as a functional description of lpr and some of its special uses. The binary code for lpr is in /bin/lpr while the source resides in /usr/source/sl/lpr.c. Attached, as Appendix A, is the appropriate page from the Reference Manual that describes the various options to lpr.

II. Functional Description

When lpr is executed, a "temporary file" (named TFxxx - where xxx are random digits) containing control card images is created in the /usr/lpd directory. The control cards include a $IDENT card used to identify the invoking userid, an F (file) control card with the name of the file to be printed, and optionally a U (Unlink) control card to tell the line printer daemon to unlink the printed file. On completion of its processing lpr changes the filename TFxxx to a DFxxx (descriptor file) to inform the line printer daemon that the files are ready for output to the printer.

The line printer daemon, /etc/lpd, looks through the /usr/lpd directory periodically for files with the name DFxxx and uses the control cards images to decide what files are to be printed.

If the user had invoked lpr with a fully qualified file-system name (e.g., lpr /usr/iaw/junk) then an F (file) control card is generated with that name (F /usr/iaw/junk) and no U (unlink) card is needed. On the other hand, if the user had specified "lpr filename", then lpr would make a link within the /usr/lpd directory, by the name LFxxx, to this file in the user's current directory. Moreover, an F card specifying /user/lpd/LFxxx and a U card specifying /usr/lpd/LFxxx would have been generated to tell the daemon to print the file pointed to by /user/lpd/LFxxx and to unlink it when the printing was completed.

When the user invokes lpr as a filter (e.g., cat filename ¦ lpr) then a copy file (CFxxx) is created in the /usr/lpd directory, and lpr builds an F card and a U card specifying /usr/lpd/CFxxx.

Options are also permitted on the lpr command to allow removing the file after printing (-) or to force lpr to copy the file before printing (+). The plus option causes a copy (CFxxx) file to be created in /usr/lpd. If the minus option is specified, lpr makes a link entry , LFxxx, in /usr/lpd and unlinks the original file. When the printing is completed the daemon will unlink the LFxxx file and cause it to be removed. Notice this means that removing a file, with rm, after executing lpr will not affect the printing since an additional link was created to the file from /usr/lpd.

III. Special Cases

There are a few special cases in the use of lpr.

A. lpr *
B. lpr directory

A. lpr *

This produces one descriptor file (DFxxx) containing two entries (File, F card and Unlink, U card) for every file in the current directory. Each F and U card points to the Link File created for each of the files in current directory.

Directories within the current directory are treated as files so that the printed output from these files is garbage. Binary files are similarly treated as ASCII files and may cause the printer to eject many pages of garbage.

For example, if the current directory has two files "pgm.c" and "writing" then the DF file created by executing lpr * might look like:

```
L$ ident m0000,m300,iaw
F/usr/lpd/lf958
U/usr/lpd/lf958
F/usr/lpd/lf710
U/usr/lpd/lf710
```

where /usr/lpd/lf958 is a link to pgm.c and /usr/lpd/lf710 is a link to writing.

B. lpr directory

Executing lpr directory treats directory as a file and the output will be garbage. Depending on how the directory name is specified, however, will result in different looking entries in the /usr/lpd directory. If the full pathname of the directory is given no Unlink (U) card is generated. If, however, the directory specified in the command is not given as a full pathname, lpr treats the directory as a file. A link to the directory is attempted but fails since links to directories are illegal. Therefore upon failure a copy file is generated. When the line printer daemon is executed the contents of the copy file (garbage) is printed.

IV. Function Description

Attached as Appendix B is the listing of lpr.c which has been commented to provide an easier analysis of the code for those users who are planning to change it. The following is a brief synopsis of what each of the six functions within lpr does.

The basic part of the lpr.c program is the MAIN function. It has two parameters: an argument count and a vector of arguments. From the arguments it is decided what type files are to be generated. A temporary file (TFxxx) is created on all initiations of the lpr command. This file will contain the control card images used by the line printer daemon. Just prior to the completion of lpr the MAIN function renames the temporary file to a descriptor file (DFxxx) by linking the two files and then unlinking the temporary file. If the temporary file has not been renamed this indicates to the line printer daemon that the copy file (CFxxx) or the link File (LFxxx) has not been completed. The MAIN function checks the argument list for a file with a full pathname. If this is the case an "F" and the pathname of the file are passed to the CARD function, which generates the file (F card) image for the descriptor file. The descriptor file is the only file output in this instance. There are only two entries in this file instead of three (the contents of the descriptor file when it points to a link or a copy file). These card images are:

```
                L$       ident[job and box number],login name
                F        pathname
```

The line printer daemon finds a DF file and prints the file pointed to by the file (F card) image. If the "+" option was specified a copy file is created. If however, the file to be printed is not specified as a full pathname and a copy file is not forced by the "+" option, the MAIN function tries to link to the file to be printed. If a link is made successfully a link file with pathname /usr/lpd/LFxxx is created and this pathname is used on the file(F card) and unlink(U card). Otherwise a copy file is created in the COPY function with pathname /usr/lpd/CFxxx and the F card and the U card use this pathname. The argument list is then checked for a "-". If this option was specified MAIN unlinks (removes) the original file. The descriptor file generated has three entries produced by the CARD function. They appear as follows:

```
                L$       ident[job and box numbers],login name
                F        /usr/lpd/CFxxx(LFxxx)
                U        /usr/lpd/CFxxx(LFxxx)
```

The line printer daemon prints the copy file (CFxxx) or the link file (LFxxx) which is found in /usr/lpd. When this is done the daemon looks for the unlink (U card) and, if present, unlinks the file specified (the copy or link file in /usr/lpd).

The COPY function is called by the MAIN function using a file descriptor as input. The file descriptor is generated by a successful open of the file to be printed. As the function name indicates a copy of the file to be printed is made. The name of the new file is /usr/lpd/CFxxx.

The CARD function generates control card images for the descriptor file. The input to this function is an L, F, and/or a U followed by a string of characters containing the L$ ident information or the pathname of the file the daemon will print. Each call to CARD generates one of the above card images and stores these images in the temporary file.

The IDENT function is invoked by the MAIN function, but has no parameters passed to it. The information for the L$ ident card is taken from the password file and passed to the CARD function. The information for this card image is obtained from the first and fourth fields of the password file for the userid invoking lpr.

The functions RANNAME and RANC work together to generate a unique file name. When a temporary, descriptor, link, or copy file is created it is given a pathname similar to

```
/usr/lpd/TFxxx(DFxxx)(LFxxx)(CFxxx)
```

respectively. This string is passed to RANNAME where a unique name is given to the file. RANC generates a random number and passes this number to RANNAME. The latter stores the number into the eleventh, twelfth, or thirteenth position of the string respectively. In this way the x's on the end of the pathname are changed to numbers. RANNAME checks the status of the new file name and returns a pointer to the calling function. The status check guarantees that the new name is unique.

These notes have attempted to document the internals of the lpr command so that users would have a better understanding of problems that arise with use of lpr or if they wished to change the code.

[illegible]

MH-8234-IAW-unix

[illegible]

I. A. Winheim

Attachments

Copy to
G. L. Baldwin
K. Brandt
J. Cloutier
W. Coston
R. Crane
J. DeFelice
P. Kennedy
G. Foley
A. G. Fraser
R. Haight
P. Hamilton
J. Jarvis
H. S. London
H. Lycklama
F. Menzel
J. Molinelli
T. O'Connell
R. Pawnyk
R. Perdue
H. Pierson
D. M. Ritchie
M. M. Rochkind
D. Sandel
J. I. Sheetz
D. Sidor
C. A. Strohecker
T. Tabloski
B. A. Tague
K. Thompson
R. White THIS COPY FOR
G. W. R. Luderer

NAME

lpr — on line print

SYNOPSIS

lpr [−] [+] [+−]file ...

DESCRIPTION

Lpr arranges to have the line printer daemon print the file arguments.

Normally, each file is printed in the state it is found when the line printer daemon reads it. If a particular file argument is preceded by **+**, or a preceding argument of **+** has been encountered, then *lpr* will make a copy for the daemon to print. If the file argument is preceded by **−**, or a preceding argument of **−** has been encountered, then *lpr* will unlink (remove) the file.

If there are no arguments, then the standard input is read and on-line printed. Thus *lpr* may be used as a filter.

FILES

/usr/lpd/*	spool area
/etc/passwd	personal ident cards
/etc/lpd	daemon

SEE ALSO

lpd (VIII), passwd (V)

BUGS

Attachment A

```
/*
 *      lpr -- on line print to Line Printer
 */

tfname;         /* pointer to the temporary file */
nact;           /* number active */
tff;            /* temporary file descriptor */
first;          /* indicates first pass */

/*
 * MAIN checks the argument list and decides what files
 * are to be generated for the line printer daemon.
 * A Temporary File(TFxxx) in /usr/lpd is created to
 * store control card images until completion of the
 * Copy File or Link File. A Link File(LFxxx) in /usr/lpd
 * is created if a link to the input file is possible.
 * A Copy File(CFxxx) in /usr/lpd is created in all other
 * instances except one. when the input file is a full
 * pathname no Copy or Link Files are necessary and the
 * Unlink control card in the Temporary File is not
 * generated (do not want daemon to unlink input file).
 * when all files are completed the Temporary File
 * name is changed to Descriptor File(DFxxx) in /usr/lpd.
 * It is this file that the line printer daemon uses to
 * print the proper file and to unlink the proper file.
 * MAIN also checks the argument list and if a "+" is
 * present a Copy File is created, if a "-" is present
 * the input file is unlinked (removed).
*/

main(argc, argv)
int argc;
char *argv[];
{
        char *arg, *remote;
        int c, f, flag, cflag;

        /*
         * Create a Temporary File
        */

        flag = 0;
        tfname = ranname("/usr/lpd/tfxxx");
        if(tfname)
                tff = creat(tfname, 0666); else
                tff = -1;
        if(tff < 0) {
                printf("Cannot create in /usr/lpd\n");
                exit();
        }


        ident();

        /*
         * if lpr is used as a filter (pipe) make a copy of the file
        */

        if(argc == 1)
                copy(0);
```

<u>Attachment B</u>

```
/*
 * check argument list of lpr command
*/

while(--argc) {
        arg = *++argv;
        c = *arg;
        if(c == '+' || c == '-') {
                cflag = c;
                if(*++arg == '\0') {
                        flag = cflag;
                        continue;
                }
        } else
                cflag = flag;

/*
 * if argument "+" then make a copy of input file
*/

        if(cflag == '+')
                goto cf;

/*
 * Check for full pathname.
*/

        if(*arg == '/' && cflag != '-') {
                card('F', arg);
                nact++;
                continue;
        }

/*
 * Check if link can be made and make proper file type.
*/

        f = rannname("/usr/lpd/lfxxx");
        if(f) {
                if(link(arg, f))
                        goto cf;
                card('F', f);
                card('U', f);
                nact++;
                goto df;
        }
cf:
        f = open(arg, 0);
        if(f < 0) {
                printf("Cannot open %s\n", arg);
                continue;
        }
        copy(f);
        close(f);

/*
 * For "-" option remove input file.
*/

df:
```

```
                if(cflag == '-') {
                        f = unlink(arg);
                        if(f < 0)
                                printf("Cannot remove %s\n", arg);
                }
        }

        /*
         * Rename Temporary File to Descriptor File , call daemon.
        */

        if(nact) {
                f = ranname("/usr/lpd/dfxxx");
                if(f)
                        link(tfname, f); else
                        printf("Cannot rename in /usr/lpd\n");
                unlink(tfname);
                execl("/etc/lpd", "lpd", 0);
        }
        unlink(tfname);
}

/*
 * COPY creates a Copy File (CFxxx) in /usr/lpd and
 * puts a copy of the contents of the input file into it.
 * To prevent excessive output the size of the file to be
 * printed is checked. Currently 400 blocks (variable nr) of
 * 512 bytes is the upper limit. The information for the
 * File (F card) and Unlink (U card) is passed to the CARD
 * function.
*/

copy(f)
int f;
{
        int fn, ff, i, nr, nc;
        static int buf[256];


        /*
         * create a copy file in /usr/lpd
        */

        fn = ranname("/usr/lpd/cfxxx");
        if(fn)
                ff = creat(fn, 0666); else
                ff = -1;
        if(ff < 0) {
                printf("Cannot create in /usr/lpd\n");
                return;
        }
        nc = 0;
        nr = 0;

        /*
         * copy contents of input file into Copy File.
        */

        while((i = read(f, buf, 512)) > 0) {
                write(ff, buf, i);
                nc =+ i;

        /*
         * check the size of the copy file.
        */

                if(nc >= 512) {
                        nc =- 512;
                        nr++;
                        if(nr > 400) {
                                printf("Copy file is too large\n");
                                break;
                        }
                }
        }

        close(ff);
        card('F', fn);
        card('U', fn);
        nact++;
}


/*
 * CARD generates control card images for the line printer
 * daemon to use. There are three types of cards:
 * Login         L $  ident[job+box],login name
 * File          F pathname of file lpd will print
 * Unlink        U pathname of file lpd will unlink
 * The Unlink card image is not always present. If a full
 * pathname is used on the lpr call lpd will print that
 * file directly an no unlink card is needed to remove
 * a Copy or Link File.Each pass creates one card
 * image which is stored in the Temporary File.
*/

card(c, s)
int c;
char s[];
{
        char *p1, *p2;
        static char buf[512];
        int col;

        p1 = buf;
        p2 = s;
        col = 0;

        /*
         * put L,F,or U in col 0
        */

        *p1++ = c;

        /*
         * Follow L,F,or U with proper information.
        */

        while((c = *p2++) != '\0') {
                *p1++ = c;
```

```
                        col++;
                }
                *p1++ = '\n';

                write(tff, buf, col+2);
        }

        /*
         * IDENT retrives the information needed for the Login control
         * card from the password file. The Userid is used to reach the
         * appropriate line in pasword. Then the 4th parameter (job and
         * box humbers if existent) are captured. The 1st parameter (login
         * name) is then taken , This information is then passed to CARD.
        */

        ident()
        {
                int c, i, j, n;
                char *b1p;
                static char b1[100], b2[100];

                b1p = b1;
                if(getpw(getuid(), b1)) {
                        b1p = "pdp::m0130,m322:";
                }

                j = 0;
                while(c = "$    ident    "[j])
                        b2[j++] = c;
                i = 0;
                n = 4;

                /*
                 * get 4th parameter of pasword file if present
                */

                while(--n) while(b1p[i++] != ':');
                while((c = b1p[i++]) != ':')
                        b2[j++] = c;
                b2[j++] = ',';

                /*
                 * get 1st parameter of password file
                */

                i = 0;
                while((c = b1p[i++]) != ':')
                        b2[j++] = c;
                b2[j++] = '\0';

                card('L', b2);
        }

        /*
         * RANNAME generates unique random names for the Copy(CFxxx),
         * Link(LFxxx),Temporary(TFxxx),and Descriptor(DFxxx) Files.
         * The xxx (11th,12th,and 13th characters in the file pathname)
         * is replaces by 3 characters returned to RANNAME by RANC. The
         * status on the new file name is checked to insure the name
                                                                    * is unique.
```

```
*/

ranname(s)
char s[];
{
        static int buf[20];


loop:
        s[11] = ranc();
        s[12] = ranc();
        s[13] = ranc();

        if(stat(s, buf))
                return(s);
        goto loop;
}

/*
 * RANC is called by RANNAME to generate random numbers using the
 * random number generator. If it is not the first pass through
 * the random number generator is re-initialize with the low word
 * of the time parameter(whose value keeps changing). The number
 * generated is divided by 10 and the remainder is converted to
 * ascii. This ascii character is then returned to RANNAME.
*/

ranc()
{
        int buf[2], c;


        if(!first) {
                time(buf);
                srand(buf[1]);
                first++;
        }

        c = rand();
        return(((c>>11)%10)+'0');
}
```