```
**********************************************************
    4004 Thu May  7 19:54:40 2015
new/usr/src/cmd/lp/lib/lp/getlist.c
5612 lpadmin dumps core in getlist
**********************************************************
```

```
/*
 * 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 2015 Gary Mills
 */

/*      Copyright (c) 1984, 1986, 1987, 1988, 1989 AT&T */
/*        All Rights Reserved   */


#ident  "%Z%%M% %I%     %E% SMI"       /* SVr4.0 1.13  */
/* EMACS_MODES: !fill, lnumb, !overwrite, !nodelete, !picture */

#include "string.h"
#include "errno.h"
#include "stdlib.h"

#include "lp.h"

#if     defined(__STDC__)
static char             *unq_strdup ( char * , char * );
#else
static char             *unq_strdup();
#endif

/**
 ** getlist() - CONSTRUCT LIST FROM STRING
 **/

/*
 * Any number of characters from "ws", or a single
 * character from "hardsep", can separate items in the list.
 */

char **
#if     defined(__STDC__)
getlist (
        char *                  str,
        char *                  ws,
        char *                  hardsep
)
```



```
#else
getlist (str, ws, hardsep)
        register char           *str,
                                *ws;
        char                    *hardsep;
#endif
{
        register char           **list,
                                *p,
                                *sep,
                                c;

        int                     n,
                                len;

        char                    buf[10];


        char                    *copy,
                                *begin;

        if (!str || !*str)
                return (0);

        /*
         * Construct in "sep" the full list of characters that
         * can separate items in the list. Avoid a "malloc()"
         * if possible.
         */
        len = strlen(ws) + strlen(hardsep) + 1;
        if (len > sizeof(buf)) {
                if (!(sep = Malloc(len))) {
                        errno = ENOMEM;
                        return (0);
                }
        } else
                sep = buf;
        strcpy (sep, hardsep);
        strcat (sep, ws);

        /*
         * Copy the input string because getlist() sometimes writes to it.
         */
        if (!(begin = Strdup(str))) {
                errno = ENOMEM;
                return (0);
        }
        copy = begin;

        /*
         * Skip leading white-space.
         */
        copy += strspn(copy, ws);
        if (!*copy) {
                Free (begin);
        str += strspn(str, ws);
        if (!*str)
                return (0);
        }

        /*
         * Strip trailing white-space.
         */
        p = strchr(copy, '\0');
        while (--p != copy && strchr(ws, *p))
        p = strchr(str, '\0');
```

```
105         while (--p != str && strchr(ws, *p))
124                 ;
125         *++p = 0;

127         /*
128          * Pass 1: Count the number of items in the list.
129          */
130         for (n = 0, p = copy; *p; ) {
112         for (n = 0, p = str; *p; ) {
131                 if ((c = *p++) == '\\')
132                         p++;
133                 else
134                         if (strchr(sep, c)) {
135                                 n++;
136                                 p += strspn(p, ws);
137                                 if (
138                                         !strchr(hardsep, c)
139                                      && strchr(hardsep, *p)
140                                 ) {
141                                         p++;
142                                         p += strspn(p, ws);
143                                 }
144                         }
145         }

147         /*
148          * Pass 2: Create the list.
149          */

151         /*
152          * Pass 1 counted the number of list separaters, so
153          * add 2 to the count (includes 1 for terminating null).
154          */
155         if (!(list = (char **)Malloc((n+2) * sizeof(char *)))) {
156                 errno = ENOMEM;
157                 goto Done;
158         }

160         /*
161          * This loop will copy all but the last item.
162          */
163         for (n = 0, p = copy; *p; )
145         for (n = 0, p = str; *p; )
164                 if ((c = *p++) == '\\')
165                         p++;
166                 else
167                         if (strchr(sep, c)) {

169                                 p[-1] = 0;
170                                 list[n++] = unq_strdup(copy, sep);
152                                 list[n++] = unq_strdup(str, sep);
171                                 p[-1] = c;

173                                 p += strspn(p, ws);
174                                 if (
175                                         !strchr(hardsep, c)
176                                      && strchr(hardsep, *p)
177                                 ) {
178                                         p++;
179                                         p += strspn(p, ws);
180                                 }
181                                 copy = p;
163                                 str = p;

183                         }
```



```
185         list[n++] = unq_strdup(copy, sep);
167         list[n++] = unq_strdup(str, sep);

187         list[n] = 0;

189 Done:   if (sep != buf)
190                 Free (sep);
191         Free (begin);
192         return (list);
193 }
```

_____*unchanged_portion_omitted_*