Materials for Bruce's mkserv code review

The files most in need of review are presented first.

os.c.diff-D

A context diff wouldn't have been adequate for this one. My changes are presented like this:

```
#ifndef __________________________________________CHANGES_BY_BRLEWIS____
        if((p = index(filesystem, ':')) == NULL) {
#else /* __________________________________________CHANGES_BY_BRLEWIS____ */
        if((p = strchr(filesystem, ':')) == NULL) {
#endif /* __________________________________________CHANGES_BY_BRLEWIS____ */
```

The file is generated by diff -D such that compiling without the CHANGES_BY_BRLEWIS symbol will yield the old; compiling with the symbol defined will produce the new.

Other changes to review can be found at the following lines:
163, 232, 271, 314, 332, 354, 408, 417, 460, 471, 548, 559

They consist of places where I've added sgi onto an #if defined() line, and added a #error line later indicating what I think the requirement is for a platform to be added into the list.

os.h.patch

I put exactly the same things for sgi as for sun.

mkserv.h.patch and util.c.patch

I changed #ifndef sun to #ifdef ultrix in reference to strdup(). In general, I think new platforms will have it, and keeping a strdup prototype in mkserv.h would only create conflicting declarations.

environment.c.patch

Since we only compile mkserv on POSIX-compliant platforms now, and POSIX requires putenv(), I changed environment.c to always use putenv with "#ifdef 1". I didn't take out the setenv() code.

everything else

The remaining patches change index(), bzero() and bcopy() to their POSIX equivalence. I made an emacs macro for the bcopy() changes, checking that there were no places that would confuse the macro, e.g. bcopy(f(x,y), a, b). The first comma after bcopy always delimited the first argument to bcopy().



```
1  #if !defined(lint) && !defined(SABER)
2  static char rcsid[] = "$Id$";
3  #endif
4
5  /* This file handles the machine specific configuration */
6
7  #include "mkserv.h"
8  #include "os.h"
9  #include <pwd.h>
10
11 #ifdef _IBMR2
12 #include <userpw.h>
13 #include <usersec.h>
14 #include <sys/id.h>
15 #endif
16
17 /*
18  * char *os_serverdir()
19  *      Returns the operating system location of OS_SERVERDIR
20  *      Static allocation - do not free or modify
21  */
22
23
24 char *
25 os_serverdir()
26 {
27         return(OS_SERVERDIR);
28 }
29
30 char *
31 os_sitedir()
32 {
33         return(OS_SITE);
34 }
35
36 char *os_versionfile()
37 {
38         struct stat statb;
39         static char versionfile[MAXPATHLEN+1];
40         static int init=0;
41
42         if(init) return versionfile;
43
44         /* Test for correct */
45         strcpy(versionfile, VERSION_FILE);
46         if(!stat(versionfile, &statb)) {
47                 init=1;
48                 return versionfile;
49         }
50         /* Test for old */
51         strcpy(versionfile, OLD_VERSION_FILE);
52         if(!stat(versionfile, &statb)) {
53                 init=1;
54                 return versionfile;
55         }
56         /* Fallback */
57         strcpy(versionfile, VERSION_FILE);
58         return versionfile;
59 }
60
61 os_create_serverdir(char *dir)
62 {
63         /* Umask handles the issues */
64         if(mkdir(dir, 0755)) {
65                 fprintf(stderr, "Unable to create %s\n", dir);
66                 return -1;
67         }
68         return 0;
69 }
70
71 /* Checks if service path is valid
72  * Returns 0 if ok, -1 if not, -2 if format wrong
73  * Handles parsing of filesystem:path
74  * Will attach if necessary
75  */
76 os_path_ok(char *name)
77 {
```

```
        char filesystem[MAXPATHLEN+1], path[MAXPATHLEN+1], *p;
        register DIR *dirp;

        if(name == NULL) return -2;

        strcpy(filesystem, name);
#ifndef ____________________________________________CHANGES_BY_BRLEWIS____
        if((p = index(filesystem, ':')) == NULL) {
#else /* ___________________________________________CHANGES_BY_BRLEWIS____ */
        if((p = strchr(filesystem, ':')) == NULL) {
#endif /* _________________________________________CHANGES_BY_BRLEWIS____ */
                /* Format wrong.... fatal */
                return -2;
        }
        *p = '\0';
        p++;
        strcpy(path, name + (p - filesystem));

        /* May need to attach */
        if(filesystem[0]) {
                if(os_attach(filesystem)) return -1;
        }

        /* Code to check..... */

        if((dirp = opendir(path)) == NULL) {
                /* Can't open direct - fatal */
                return -1;
        }
        if(readdir(dirp) == NULL) {
                /* There has to be at least '.' */
                closedir(dirp);
                return -1;
        }
        closedir(dirp);

        /* Path looks ok */
        return 0;
}

os_attach(char *filesys)
{
        char buf[MAXPATHLEN+1];
        sprintf(buf, "/bin/athena/attach -q -n -h %s\n", filesys);
        return(system(buf));
}

char *os_start_path(struct local_env *lenv)
{
        static char buf[MAXPATHLEN+1];
        char buf1[30], *p;

        strcpy(buf1, lenv->vers);
        p = buf1;
        while(*p && (isdigit(*p) || *p == '.')) p++;
        *p = '\0';

        sprintf(buf, "/mit/mkserv/services/%s", buf1);

        return buf;
}

char *os_platform()
{
        return (OS_PLATFORM);
}

void os_run_script(struct service *s, int d)
{
        char buf[MAXPATHLEN+1];

        sprintf(buf, "%s/%s%s", s->path, s->name, service_suffix(d));
        if(s->fh[d].version)
                printf("\nRunning %s (%s)\n", buf, s->fh[d].version);
        else
                printf("\nRunning %s\n", buf);
        system(buf);
```



```
}

/* char *os_switch_service(char *line, char *on)
 * Switches service. Returns a poitiner to a new line after freeing the old
 * Returns NULL if unable to do so
 */
char *os_switch_service(char *line, char *on)
{
#ifndef ____________________________________________CHANGES_BY_BRLEWIS____
#if defined(ultrix) || defined(ibm032) || defined(vax) || defined(_AIX) || define
+d(sun)
#else /* ___________________________________________CHANGES_BY_BRLEWIS____ */
#if defined(ultrix) || defined(ibm032) || defined(vax) || defined(_AIX) || define
+d(sun) || defined(sgi)
#endif /* _________________________________________CHANGES_BY_BRLEWIS____ */
        int count=0, ingap;
        char buf[200];
        char *p, *newstring;
        int force = 1;

#if defined(_AIX)
        newstring="on";
        if(!strcmp(on, "disable"))
            newstring="off";
#else
        newstring="unswitched";
#endif
        if(!strcmp(on, "on")) {
                newstring="switched";
                force = 0;
        }
        /* swon = 1-> switched entry */

        /* First get to data point */
        /* Format: "service stream tcp nowait {switched/unswitched}" */
        p = line;
        ingap = 0;
        while(*p && count < 4) {
                if(*p == ' ' || *p == '\t') {
                        ingap = 1;
                } else {
                        if(ingap == 1) {
                                ingap = 0;
                                count ++;
                        }
                }
                if(count < 4) p++;
        }

        if(!*p) return NULL;
        /* Now pointing at switched or unswitched */
        if(!strncmp(p, newstring, strlen(newstring)) && (force == 0)) {
                return line;
        }

#if 0
        /* We don't do this - some services by default are not enabled */
        /* Need to make a substitution */
        if(line[0] == '#') {
                strcpy(buf, line + 1);
                buf[p-line+1] = '\0';
        }
        else {
#endif
                strcpy(buf, line);
                buf[p-line]='\0';
#if 0
        }
#endif
        strcat(buf, newstring);
        ingap = 0;
        count = 0;
        while(*p && isalpha(*p)) p++;
        if(!*p) return NULL;
        strcat(buf, p);
        p = strdup(buf);
        if(p) free(line);
```

```
        return p;
#else /* ultrix, ibm032, vax, aix */
#ifndef ___________________________________________CHANGES_BY_BRLEWIS____
#error Need configuration
#else /* ___________________________________________CHANGES_BY_BRLEWIS____ */
#error If this platform uses Athena inetd, just add it to list.
#endif /* ___________________________________________CHANGES_BY_BRLEWIS____ */
#endif
}

/* The next two routines are stolen from hesiod/hespwnam.c */
/* Move the pointer forward to the next colon-separated field in the
 * password entry.
 */

static char *
_NextPWField(ptr)
char *ptr;
{
        while (*ptr && *ptr != '\n' && *ptr != ':')
                ptr++;
        if (*ptr)
                *ptr++ = '\0';
        return(ptr);
}

struct passwd * os_convert_hesiod_to_pwd(char *line)
{
        static char buf[256];
        static struct passwd pw_entry;

        register char *p;
        (void) strcpy(buf, line);
        p = buf;
        pw_entry.pw_name = p;
        p = _NextPWField(p);
        pw_entry.pw_passwd = p;
        p = _NextPWField(p);
        pw_entry.pw_uid = atoi(p);
        p = _NextPWField(p);
        pw_entry.pw_gid = atoi(p);
#ifndef ___________________________________________CHANGES_BY_BRLEWIS____
#if (!defined(_AIX) || (AIXV < 31)) && !defined(sun)
#else /* ___________________________________________CHANGES_BY_BRLEWIS____ */
#if 0 /* POSIX does not have pw_quota */
#endif /* ___________________________________________CHANGES_BY_BRLEWIS____ */
        pw_entry.pw_quota = 0;
#if defined(_AIX) && (AIXV < 31)
        pw_entry.pw_age =
#endif
        pw_entry.pw_comment = "";
#endif
        p = _NextPWField(p);
        pw_entry.pw_gecos = p;
        p = _NextPWField(p);
        pw_entry.pw_dir = p;
        p = _NextPWField(p);
        pw_entry.pw_shell = p;
        while (*p && *p != '\n')
                p++;
        *p = '\0';
        return(&pw_entry);
}

static lock_passwd()
{
        int cnt, fd;

        cnt = 10;
        while (cnt-- > 0 &&
               (fd = open("/etc/ptmp", O_WRONLY|O_CREAT|O_EXCL, 0644)) < 0)
                sleep(1);
        if (fd < 0) {
                printf("Failed to lock passwd\n");
        }
        (void) close(fd);
}
```



```

static unlock_passwd()
{
        (void) unlink("/etc/ptmp");
}
os_add_to_passwd (struct passwd *pwd)
{
#ifndef ___________________________________________CHANGES_BY_BRLEWIS____
#if defined(ultrix) || defined(vax) || defined(ibm032) || defined(sun)
#else /* ___________________________________________CHANGES_BY_BRLEWIS____ */
#if defined(ultrix) || defined(vax) || defined(ibm032) || defined(sun) || defined
+(sgi)
#endif /* ___________________________________________CHANGES_BY_BRLEWIS____ */
        FILE *pfile;
        struct stat statb;

        /* Handle /etc/passwd and /etc/passwd.local */
        if(stat("/etc/passwd.local", &statb)) {
                /* passwd.local doesn't exist */
                /* copy /etc/passwd to /etc/passwd.local */
                printf("Copying /etc/passwd to /etc/passwd.local\n");
                if(copy_file("/etc/passwd", "/etc/passwd.local")) {
                        perror("Copying passwd.local");
                        unlink("/etc/passwd.local");
                }
        }
#ifndef ___________________________________________CHANGES_BY_BRLEWIS____
#ifndef sun
#else /* ___________________________________________CHANGES_BY_BRLEWIS____ */
#ifdef _AIX
#endif /* ___________________________________________CHANGES_BY_BRLEWIS____ */
        setpwfile("/etc/passwd.local");
#endif
        if(getpwnam(pwd->pw_name) == NULL) {
                lock_passwd();
                if((pfile=fopen("/etc/passwd.local", "a")) != NULL) {
                        fprintf(pfile, "%s:%s:%d:%d:%s:%s:%s\n",
                                pwd->pw_name,
                                pwd->pw_passwd,
                                pwd->pw_uid,
                                pwd->pw_gid,
                                pwd->pw_gecos,
                                pwd->pw_dir,
                                pwd->pw_shell);
                        fclose(pfile);
                }
                unlock_passwd();
        }
#ifndef ___________________________________________CHANGES_BY_BRLEWIS____
#ifndef sun
#else /* ___________________________________________CHANGES_BY_BRLEWIS____ */
#ifdef _AIX
#endif /* ___________________________________________CHANGES_BY_BRLEWIS____ */
        setpwfile("/etc/passwd");
#endif
        if(getpwnam(pwd->pw_name) == NULL) {
                lock_passwd();
                if((pfile=fopen("/etc/passwd", "a")) != NULL) {
                        fprintf(pfile, "%s:%s:%d:%d:%s:%s:%s\n",
                                pwd->pw_name,
                                pwd->pw_passwd,
                                pwd->pw_uid,
                                pwd->pw_gid,
                                pwd->pw_gecos,
                                pwd->pw_dir,
                                pwd->pw_shell);
                        fclose(pfile);
                }
                unlock_passwd();
        }
        return 0;
#else
#ifdef _IBMR2
        struct userpw pw_stuff;

        if(getpwnam(pwd->pw_name) == NULL) {
                putuserattr(pwd->pw_name,(char *)NULL,((void *) 0),SEC_NEW);
```

```
                putuserattr(pwd->pw_name,S_ID,(void *)(pwd->pw_uid),SEC_INT);
                putuserattr(pwd->pw_name,S_PWD,(void *)"!",SEC_CHAR);
                putuserattr(pwd->pw_name,S_PGRP,(void *)"mit",SEC_CHAR);
                putuserattr(pwd->pw_name,S_HOME,(void *)(pwd->pw_dir),SEC_CHAR);
                putuserattr(pwd->pw_name,S_SHELL,(void *)(pwd->pw_shell),SEC_CHAR);
                putuserattr(pwd->pw_name,S_GECOS,(void *)(pwd->pw_gecos),SEC_CHAR);
                putuserattr(pwd->pw_name,S_LOGINCHK,(void *)1,SEC_BOOL);
                putuserattr(pwd->pw_name,S_SUCHK,(void *)1,SEC_BOOL);
                putuserattr(pwd->pw_name,S_RLOGINCHK,(void *)1,SEC_BOOL);
                putuserattr(pwd->pw_name,S_ADMIN,(void *)0,SEC_BOOL);
                /* putuserattr(p->pw_name,"athena_temp",(void *)1,SEC_INT);*/
                putuserattr(pwd->pw_name,(char *)0,((void *) 0),SEC_COMMIT);

                enduserdb();

                setpwdb(S_READ|S_WRITE);
                strncpy(pw_stuff.upw_name,pwd->pw_name,PW_NAMELEN);
                pw_stuff.upw_passwd = pwd->pw_passwd;
                pw_stuff.upw_flags = 0;
                pw_stuff.upw_lastupdate = 0;
                putuserpw(&pw_stuff);
                endpwdb();
        }
        return 0;
#else
#ifndef __________________________________________CHANGES_BY_BRLEWIS____
#error Need configuration
#else /* __________________________________________CHANGES_BY_BRLEWIS____ */
#error If passwd file for this platform is standard format, just add to list
#endif /* __________________________________________CHANGES_BY_BRLEWIS____ */
#endif /* _IBMR2*/
#endif
}

#ifndef __________________________________________CHANGES_BY_BRLEWIS____
#if defined(vax) || defined(ultrix) || defined(ibm032) || defined(sun)
#else /* __________________________________________CHANGES_BY_BRLEWIS____ */
#if defined(vax) || defined(ultrix) || defined(ibm032) || defined(sun) || defined(sgi)
#endif /* __________________________________________CHANGES_BY_BRLEWIS____ */
static void os_remove_pentry(char *name, char *file)
{
    FILE *newfile;
    struct passwd *copypw;
    struct stat statb;
    int cnt, fd;

    cnt = 10;
    while (cnt-- > 0 &&
           (fd = open("/etc/ptmp", O_WRONLY|O_CREAT|O_EXCL, 0644)) < 0)
      sleep(1);
    if (fd < 0) {
        fprintf(stderr, "Failed to remove passwd entry %s\n", name);
        return;
    }

    if ((newfile = fdopen(fd, "w")) != NULL) {
            setpwent();
            while ((copypw = getpwent()) != 0)
                    if (strcmp(copypw->pw_name, name))
                            fprintf(newfile, "%s:%s:%d:%d:%s:%s:%s\n",
                                    copypw->pw_name,
                                    copypw->pw_passwd,
                                    copypw->pw_uid,
                                    copypw->pw_gid,
                                    copypw->pw_gecos,
                                    copypw->pw_dir,
                                    copypw->pw_shell);
            endpwent();
            fclose(newfile);
            if (stat("/etc/ptmp", &statb) != 0 || statb.st_size < 80) {
                    fprintf(stderr, "Something stomped on /etc/ptmp during removal of passwd entry\n");
            } else
                    rename("/etc/ptmp", file);
```



```
    }
}
os_remove_passwd(char *name)
{
#ifndef __________________________________________CHANGES_BY_BRLEWIS____
#ifndef sun
#else /* __________________________________________CHANGES_BY_BRLEWIS____ */
#ifdef _AIX
#endif /* __________________________________________CHANGES_BY_BRLEWIS____ */
        setpwfile("/etc/passwd.local");
#endif
        if(getpwnam(name) != NULL) {
                os_remove_pentry(name, "/etc/passwd.local");
        }

#ifndef __________________________________________CHANGES_BY_BRLEWIS____
#ifndef sun
#else /* __________________________________________CHANGES_BY_BRLEWIS____ */
#ifdef _AIX
#endif /* __________________________________________CHANGES_BY_BRLEWIS____ */
        setpwfile("/etc/passwd");
#endif
        if(getpwnam(name) != NULL) {
                os_remove_pentry(name, "/etc/passwd");
        }
        return 0;
}
#else
#ifdef _IBMR2
os_remove_passwd(char *name)
{
    static char *empty = "\0";
    char *grp, *usr;
    int i;

    if(getpwnam(name) != NULL) {
            setuserdb(S_READ|S_WRITE);

            /* Decrement reference count on user's temporary groups */
            if (getuserattr(name, S_GROUPS, (void *)&grp, SEC_LIST) == 0) {
                    while (*grp) {
                            if (getgroupattr(grp, "athena_temp", (void *)&i, SEC_INT) == 0) {
                                    if (--i > 0) {
                                            putgroupattr(grp, "athena_temp",
                                                         (void *)i, SEC_INT);
                                            putgroupattr(grp, (char *)0,
                                                         (void *)0, SEC_COMMIT);
                                    } else {
                                            putgroupattr(grp, S_USERS,
                                                         (void *)empty, SEC_LIST);
                                            putgroupattr(grp, (char *)0,
                                                         (void *)0, SEC_COMMIT);
                                            rmufile(grp, 0, GROUP_TABLE);
                                    }
                            }
                            while(*grp) grp++;
                            grp++;
                    }
            }

            /* Decrement reference count on temporary users */
            if (getuserattr(name, "athena_temp", (void *)&i, SEC_INT) == 0) {
                    if (--i > 0) {
                            putuserattr(name, "athena_temp", (void *)i, SEC_INT);
                            putuserattr(name, (char *)0, (void *)0, SEC_COMMIT);
                    } else {
                            putuserattr(name, S_GROUPS, (void *)empty, SEC_LIST);
                            putuserattr(name, (char *)0, (void *)0, SEC_COMMIT);
                            rmufile(name, 1, USER_TABLE);
                    }
            }

            /* Remove any empty temporary groups */
            grp = nextgroup(S_LOCAL, 0);
            while (grp) {
```

```
                    if (getgroupattr(grp, "athena_temp",
                                     (void *)&i, SEC_INT) == 0) {
                            if ((getgroupattr(grp, S_USERS,
                                              (void *)&usr, SEC_LIST) == -1) ||
                                *usr == 0)
                            {
                                    rmufile(grp, 0, GROUP_TABLE);
                            }
                    }
                    grp = nextgroup(0, 0);
            }

            enduserdb();
    }
    return 0;
}
#else
#ifndef __________________________________________CHANGES_BY_BRLEWIS____
Need confirguartion
#else /* __________________________________________CHANGES_BY_BRLEWIS____ */
#error If passwd file for this platform is standard format, just add to list
#endif /* __________________________________________CHANGES_BY_BRLEWIS____ */
#endif /* IBMR2 */
#endif /* os_remove_passwd */

/* Converts a BSD style inetd entry to machine specific */
char *os_convert_service_to_inetd(char *s)
{
#ifndef __________________________________________CHANGES_BY_BRLEWIS____
#if defined(vax) || defined(ibm032) || defined(ultrix) || defined(_AIX) || define
+d(sun)
#else /* __________________________________________CHANGES_BY_BRLEWIS____ */
#if defined(vax) || defined(ibm032) || defined(ultrix) || defined(_AIX) || define
+d(sun) || defined(sgi)
#endif /* __________________________________________CHANGES_BY_BRLEWIS____ */
        return(s);
#else
#ifndef __________________________________________CHANGES_BY_BRLEWIS____
#error Need configuration
#else /* __________________________________________CHANGES_BY_BRLEWIS____ */
#error If this platform uses Athena inetd, just add to list
#endif /* __________________________________________CHANGES_BY_BRLEWIS____ */
#endif
}

os_usr_not_on_root()
{
  return(OS_USR_NOT_ON_ROOT);
}

os_srvd_separate()
{
  struct stat sbuf;
  return (!stat(OS_EXISTS, &sbuf));
}
```



```
*** 1.8 1995/11/07 22:09:45
--- os.h        1996/08/19 21:25:11
***************
*** 1,3 ****
--- 1,6 ----
+ #ifndef _MKSERV_OS_H
+ #define _MKSERV_OS_H
+
  #ifdef vax
  #define OS_PLATFORM "vax"
  #define OS_SERVERDIR "/site/server"
***************
*** 29,34 ****
--- 32,43 ----
  #define OS_USR_SUBDIRS_LOCAL /* /usr/bin, et cetera are on local disk */
  #endif

+ #ifdef sgi
+ #define OS_PLATFORM "sgi"
+ #define OS_USR_NOT_ON_ROOT 1
+ #define OS_USR_SUBDIRS_LOCAL /* /usr/bin, et cetera are on local disk */
+ #endif
+
  #define VERSION_FILE "/etc/athena/version"
  #define OLD_VERSION_FILE "/etc/version"

***************
*** 48,50 ****
--- 57,61 ----
  #ifndef OS_PLATFORM
  #error Platform not defined in os.h
  #endif
+
+ #endif /* _MKSERV_OS_H */
```

```
*** 1.6 1996/06/13 22:03:25
--- 1.7 1996/08/19 21:19:03
***************
*** 147,153 ****

  extern char *find_in_string _AP((Const char *, char *));

! #ifndef sun
  extern char *strdup _AP((char *));
  #endif

--- 147,153 ----

  extern char *find_in_string _AP((Const char *, char *));

! #ifdef ultrix
  extern char *strdup _AP((char *));
  #endif

```



```
*** 1.3 1993/10/07 14:10:09
--- 1.4 1996/08/19 21:19:26
***************
*** 24,30 ****
        return NULL;
  }

! #ifndef sun
  char *
  strdup(char *s)
  {
--- 24,30 ----
        return NULL;
  }

! #ifdef ultrix
  char *
  strdup(char *s)
  {
```

```
*** 1.5 1996/06/13 22:02:29
--- 1.7 1996/09/05 18:01:55
***************
*** 15,21 ****

  insert_env(char *name, char *value)
  {
! #if defined(_IBMR2) || defined (sun)
        char buf[200], *p;
        sprintf(buf, "%s=%s", name, value);
        p = strdup(buf);
--- 15,21 ----

  insert_env(char *name, char *value)
  {
! #if 1 /* POSIX requires putenv */
        char buf[200], *p;
        sprintf(buf, "%s=%s", name, value);
        p = strdup(buf);
***************
*** 41,47 ****
        char path[MAXPATHLEN+1];
        char buf[BUFSIZ];

!       bzero(env, sizeof(struct local_env));

        strcpy(path, "/bin/athena:/usr/athena/bin:/usr/athena/etc:/usr/ucb:/bin:/
+usr/bin:/etc:/usr/etc:");

--- 41,47 ----
        char path[MAXPATHLEN+1];
        char buf[BUFSIZ];

!       memset(env, 0, sizeof(struct local_env));

        strcpy(path, "/bin/athena:/usr/athena/bin:/usr/athena/etc:/usr/ucb:/bin:/
+usr/bin:/etc:/usr/etc:");

```



```
*** 1.2 1992/05/09 22:57:51
--- 1.3 1996/09/05 18:01:55
***************
*** 17,23 ****
        int del;
        struct cmd_service *s, *s1, *ps;

!       bzero(cmd, sizeof(struct cmd_line));
        cmd->progname = strdup(argv[0]);
        if(argc == 1) {
                return -1;
--- 17,23 ----
        int del;
        struct cmd_service *s, *s1, *ps;

!       memset(cmd, 0, sizeof(struct cmd_line));
        cmd->progname = strdup(argv[0]);
        if(argc == 1) {
                return -1;
***************
*** 70,76 ****
                        if((cmd->services = (struct cmd_service *)
                                malloc(sizeof(struct cmd_service))) == NULL)
                                return -2;
!                       bzero(cmd->services, sizeof(struct cmd_service));
                        if((cmd->services->sname = strdup(p)) == NULL)
                                return -2;
                        cmd->services->del = del;
--- 70,76 ----
                        if((cmd->services = (struct cmd_service *)
                                malloc(sizeof(struct cmd_service))) == NULL)
                                return -2;
!                       memset(cmd->services, 0, sizeof(struct cmd_service));
                        if((cmd->services->sname = strdup(p)) == NULL)
                                return -2;
                        cmd->services->del = del;
***************
*** 100,106 ****
                                                 (struct cmd_service *)
                                                 malloc(sizeof(struct cmd_servi
+ce))) == NULL)
                                                        return -2;
!                                               bzero(s->next, sizeof(struct cmd_
+service));
                                                s = s->next;
                                                if((s->sname = strdup(p)) == NULL
+)
                                                        return -2;
--- 100,106 ----
                                                 (struct cmd_service *)
                                                 malloc(sizeof(struct cmd_servi
+ce))) == NULL)
                                                        return -2;
!                                               memset(s->next, 0, sizeof(struct
+cmd_service));
                                                s = s->next;
                                                if((s->sname = strdup(p)) == NULL
+)
                                                        return -2;
```

```
*** 1.5 1994/06/03 17:44:40
--- 1.7 1996/09/05 18:01:55
***************
*** 58,64 ****
        struct configs conf;
        int i;

!       bzero(&conf, sizeof(conf));

        if((f = fopen(name, "r")) == NULL) {
                fprintf(stderr, "Unable to open %s for configuration changes\n",
+name);
--- 58,64 ----
        struct configs conf;
        int i;

!       memset(&conf, 0, sizeof(conf));

        if((f = fopen(name, "r")) == NULL) {
                fprintf(stderr, "Unable to open %s for configuration changes\n",
+name);
***************
*** 81,87 ****
                                fprintf(stderr, "Out of memory in config_services
+\n");
                                return -1;
                        }
!                       bzero(l, sizeof(struct line));
                        l->data = strdup(buf);
                        if(!l->data) {
                                fprintf(stderr, "Out of memory in config_services
+\n");
--- 81,87 ----
                                fprintf(stderr, "Out of memory in config_services
+\n");
                                return -1;
                        }
!                       memset(l, 0, sizeof(struct line));
                        l->data = strdup(buf);
                        if(!l->data) {
                                fprintf(stderr, "Out of memory in config_services
+\n");
***************
*** 88,94 ****
                                return -1;
                        }
                        /* Process primary and secondary info */
!                       p = index(buf, ' ');
                        if(p) {
                                *p = '\0';
                                p++;
--- 88,94 ----
                                return -1;
                        }
                        /* Process primary and secondary info */
!                       p = strchr(buf, ' ');
                        if(p) {
                                *p = '\0';
                                p++;
***************
*** 99,105 ****
                        }
                        if(p) {
                                /* Get secondary */
!                               p1 = index(p, ' ');
                                if(p1) {
                                        p = ++p1;
                                        while(*p && !isspace(*p)) p++;
--- 99,105 ----
                        }
                        if(p) {
                                /* Get secondary */
!                               p1 = strchr(p, ' ');
                                if(p1) {
                                        p = ++p1;
                                        while(*p && !isspace(*p)) p++;
***************
```



```
*** 163,169 ****
  {
        char *p;
        char buf[1024];
!       if((p = index(lin->data, ' ')) == NULL) {
                fprintf(stderr, "Error in parsing line in deluser\n");
                return -1;
        }
--- 163,169 ----
  {
        char *p;
        char buf[1024];
!       if((p = strchr(lin->data, ' ')) == NULL) {
                fprintf(stderr, "Error in parsing line in deluser\n");
                return -1;
        }
***************
*** 181,192 ****
        struct passwd *pwd;

        /* We need to know if hesiod or not */
!       if((p = index(lin->data, ' ')) == NULL) {
                fprintf(stderr, "Error in parsing line in adduser\n");
                return -1;
        }
        strcpy(buf, ++p);
!       if((p = index(buf, ' ')) == NULL) {
                fprintf(stderr, "Error in parsing line in adduser\n");
                return -1;
        }
--- 181,192 ----
        struct passwd *pwd;

        /* We need to know if hesiod or not */
!       if((p = strchr(lin->data, ' ')) == NULL) {
                fprintf(stderr, "Error in parsing line in adduser\n");
                return -1;
        }
        strcpy(buf, ++p);
!       if((p = strchr(buf, ' ')) == NULL) {
                fprintf(stderr, "Error in parsing line in adduser\n");
                return -1;
        }
***************
*** 232,243 ****
        }

        /* We need the service name */
!       if((p = index(lin->data, ' ')) == NULL) {
                fprintf(stderr, "Error in parsing line in delervice\n");
                return -1;
        }
        strcpy(buf, ++p);
!       if((p = index(buf, ' ')) != NULL) {
          *p = '\0';
        }

--- 232,243 ----
        }

        /* We need the service name */
!       if((p = strchr(lin->data, ' ')) == NULL) {
                fprintf(stderr, "Error in parsing line in delervice\n");
                return -1;
        }
        strcpy(buf, ++p);
!       if((p = strchr(buf, ' ')) != NULL) {
          *p = '\0';
        }

***************
*** 282,294 ****
        }

        /* We need the service name */
!       if((p = index(lin->data, ' ')) == NULL) {
                fprintf(stderr, "Error in parsing line in addservice\n");
```

```
                return -1;
        }
        strcpy(buf, ++p);
        p1 = p;
!       if(((p = index(buf, ' ')) == NULL) && ((p = index(buf, '\t')) == NULL)) {
                fprintf(stderr, "Error in parsing line in addservice\n");
                return -1;
        }
--- 282,294 ----
        }

        /* We need the service name */
!       if((p = strchr(lin->data, ' ')) == NULL) {
                fprintf(stderr, "Error in parsing line in addservice\n");
                return -1;
        }
        strcpy(buf, ++p);
        p1 = p;
!       if(((p = strchr(buf, ' ')) == NULL) && ((p = strchr(buf, '\t')) == NULL))
+ {
                fprintf(stderr, "Error in parsing line in addservice\n");
                return -1;
        }
***************
*** 357,368 ****
                        return -1;
        }
        /* We need the service name */
!       if((p = index(lin->data, ' ')) == NULL) {
                fprintf(stderr, "Error in parsing line in switchservice\n");
                return -1;
        }
        strcpy(buf, ++p);
!       if((p = index(buf, ' ')) == NULL) {
                fprintf(stderr, "Error in parsing line in switchservice\n");
                return -1;
        }
--- 357,368 ----
                        return -1;
        }
        /* We need the service name */
!       if((p = strchr(lin->data, ' ')) == NULL) {
                fprintf(stderr, "Error in parsing line in switchservice\n");
                return -1;
        }
        strcpy(buf, ++p);
!       if((p = strchr(buf, ' ')) == NULL) {
                fprintf(stderr, "Error in parsing line in switchservice\n");
                return -1;
        }
***************
*** 406,417 ****
        }

        /* We need the variable name */
!       if((p = index(lin->data, ' ')) == NULL) {
                fprintf(stderr, "Error in parsing line in conf\n");
                return -1;
        }
        strcpy(buf, ++p);
!       if((p = index(buf, ' ')) == NULL) {
                fprintf(stderr, "Error in parsing line in conf\n");
                return -1;
        }
--- 406,417 ----
        }

        /* We need the variable name */
!       if((p = strchr(lin->data, ' ')) == NULL) {
                fprintf(stderr, "Error in parsing line in conf\n");
                return -1;
        }
        strcpy(buf, ++p);
!       if((p = strchr(buf, ' ')) == NULL) {
                fprintf(stderr, "Error in parsing line in conf\n");
                return -1;
        }
```



```
***************
*** 439,451 ****
                 * After = replace text up to ;
                 */
                strcpy(buf, l->data);
!               if((p = index(buf, '=')) == NULL) {
                        fprintf(stderr, "Error in parsing line in rc.conf\n");
                        return -1;
                }
                p++; /* p points to beggining of data for buf */
                *p = '\0';
!               if((p1 = index(l->data+ (p-buf), ';')) == NULL) {
                        fprintf(stderr, "Error in parsing line in rc.conf\n");
                        return -1;
                }
--- 439,451 ----
                 * After = replace text up to ;
                 */
                strcpy(buf, l->data);
!               if((p = strchr(buf, '=')) == NULL) {
                        fprintf(stderr, "Error in parsing line in rc.conf\n");
                        return -1;
                }
                p++; /* p points to beggining of data for buf */
                *p = '\0';
!               if((p1 = strchr(l->data+ (p-buf), ';')) == NULL) {
                        fprintf(stderr, "Error in parsing line in rc.conf\n");
                        return -1;
                }
```

```
*** 1.2 1992/05/09 22:57:59
--- 1.3 1996/09/05 18:01:55
***************
*** 213,223 ****
        char b[BUFSIZ];

        if((f = fopen(name, "r")) == NULL) return errno;
!       bzero(b, BUFSIZ);

        while(fgets(b, BUFSIZ, f) != NULL) {
                /* Very inefficient but works */
!               bcopy(b, buf, BUFSIZ);
        }

        (void) fclose(f);
--- 213,223 ----
        char b[BUFSIZ];

        if((f = fopen(name, "r")) == NULL) return errno;
!       memset(b, 0, BUFSIZ);

        while(fgets(b, BUFSIZ, f) != NULL) {
                /* Very inefficient but works */
!               memcpy(buf, b, BUFSIZ);
        }

        (void) fclose(f);
```



```
*** 1.8 1996/06/13 22:03:20
--- 1.9 1996/08/19 20:53:08
***************
*** 215,221 ****
        sp = current_services;
        while(sp) {
                if(!find_service(sp->name, new_services) &&
!                  !index(sp->name, ':')) {
                        if(sp->exists[S_DEL]) {
                                os_run_script(sp, S_DEL);
                        }
--- 215,221 ----
        sp = current_services;
        while(sp) {
                if(!find_service(sp->name, new_services) &&
!                  !strchr(sp->name, ':')) {
                        if(sp->exists[S_DEL]) {
                                os_run_script(sp, S_DEL);
                        }
***************
*** 226,232 ****
        /* Now the additions */
        sp = new_services;
        while(sp) {
!               if(!index(sp->name, ':')) {
                        if(sp->exists[S_ADD]) {
                                os_run_script(sp, S_ADD);
                        }
--- 226,232 ----
        /* Now the additions */
        sp = new_services;
        while(sp) {
!               if(!strchr(sp->name, ':')) {
                        if(sp->exists[S_ADD]) {
                                os_run_script(sp, S_ADD);
                        }
```

```
*** 1.3 1992/05/09 22:58:24
--- 1.5 1996/09/05 18:01:55
***************
*** 46,52 ****
        int i=0;
        char *p1;

!       bzero(&s, sizeof(s));
        s.name = strdup(serv->name);

        p = path;
--- 46,52 ----
        int i=0;
        char *p1;

!       memset(&s, 0, sizeof(s));
        s.name = strdup(serv->name);

        p = path;
***************
*** 75,81 ****

        if(found) {
                for(i = 0; i < NSERVICES; i++) {
!                       bcopy(&s.fh[i], &serv->fh[i], sizeof(struct file_header))
+;
                        serv->exists[i] = s.exists[i];
                }
                serv->path = s.path;
--- 75,81 ----

        if(found) {
                for(i = 0; i < NSERVICES; i++) {
!                       memcpy(&serv->fh[i], &s.fh[i], sizeof(struct file_header)
+);
                        serv->exists[i] = s.exists[i];
                }
                serv->path = s.path;
***************
*** 238,244 ****
                        }
                        s = s->next;
                }
!               bzero(s, sizeof(struct service));
                if((s->name = strdup(buf)) == NULL) {
                        (void) fclose(f);
                        return -1;
--- 238,244 ----
                        }
                        s = s->next;
                }
!               memset(s, 0, sizeof(struct service));
                if((s->name = strdup(buf)) == NULL) {
                        (void) fclose(f);
                        return -1;
***************
*** 264,270 ****

        s = (struct service *) malloc(sizeof(struct service));
        if(s) {
!               bzero(s, sizeof(struct service));
                if((s->name = strdup(name)) == NULL) {
                        free(s);
                        s = NULL;
--- 264,270 ----

        s = (struct service *) malloc(sizeof(struct service));
        if(s) {
!               memset(s, 0, sizeof(struct service));
                if((s->name = strdup(name)) == NULL) {
                        free(s);
                        s = NULL;
***************
*** 301,307 ****
        s = serv;
        while(s) {
                s->configured = 0;
```



```
!               if((p = index(s->name, ':')) != NULL) {
                        if(os_path_ok(s->name)) return -1;
                        if(append_service_path(&path, ++p)) return -1;
                        s->configured = 1;
--- 301,307 ----
        s = serv;
        while(s) {
                s->configured = 0;
!               if((p = strchr(s->name, ':')) != NULL) {
                        if(os_path_ok(s->name)) return -1;
                        if(append_service_path(&path, ++p)) return -1;
                        s->configured = 1;
***************
*** 312,318 ****
        /* Lookup services now */
        s = serv;
        while(s) {
!               if(index(s->name, ':') == NULL) {
                        /* Real service */
                        if(!get_service(s, path) && !check_platform(s)) {
                                s->configured = 1;
--- 312,318 ----
        /* Lookup services now */
        s = serv;
        while(s) {
!               if(strchr(s->name, ':') == NULL) {
                        /* Real service */
                        if(!get_service(s, path) && !check_platform(s)) {
                                s->configured = 1;
***************
*** 359,365 ****
                nserv = 0;
                while(s) {
                        nserv++;
!                       if(!index(s->name, ':')) {
                                strcpy(buf, "#define ");
                                strcat(buf, s->name);
                                strcat(buf, " 1\n");
--- 359,365 ----
                nserv = 0;
                while(s) {
                        nserv++;
!                       if(!strchr(s->name, ':')) {
                                strcpy(buf, "#define ");
                                strcat(buf, s->name);
                                strcat(buf, " 1\n");
***************
*** 367,373 ****
                                        (void) close(out_fd);
                                        return -1;
                                }
!                       } /* index */
                        s = s->next;

                }
--- 367,373 ----
                                        (void) close(out_fd);
                                        return -1;
                                }
!                       } /* strchr */
                        s = s->next;

                }
```

```
*** 1.5 1996/06/25 00:00:59
--- 1.6 1996/09/05 17:41:20
***************
*** 115,121 ****
        /* Phase 2 - Define service names */
        s = serv;
        while(s) {
!               if(!index(s->name, ':'))
                        fprintf(f, "#define %s 1\n", s->name);
                s = s->next;
        }
--- 115,121 ----
        /* Phase 2 - Define service names */
        s = serv;
        while(s) {
!               if(!strchr(s->name, ':'))
                        fprintf(f, "#define %s 1\n", s->name);
                s = s->next;
        }
```