UnixWorld Online: Tutorial Article No. 007 Listings

Listing 1. `Cron`-table files and utility scripts.

A. `Cron`-table file for the server:

```
# cron-table file for the server
# append this to the default root crontab file

# automates the creation of the network
# passwd files at 1AM every Monday and Thursday
0 1 * * 1,4 /usr2/sysop/upass

# purge the network passwd files at 5AM every Monday and Thursday
0 5 * * 1,4  /usr2/sysop/upass.srv

# end of script ... save as crontabfile.srv
```

B. `Cron`-table file for the client:

```
# cron-table file for the client
# append this to the default root crontab file

# updates the client passwd files at 3AM every monday and thursday
0 3 * * 1,4 /usr2/sysop/upass.cli

# end of script ... save as crontabfile.cli
```

C. Script to add a user:

```
# @(#) adduser  Add user script
Usage="Usage: $0 account-name user-id-number (>6000)"

case $# in
    2)  name=$1 ; uid=$2
        if [ $uid -le 6000 ]; then
            echo "User-id number must be greater than 6000" >&2
            exit 1
        fi ;;
    *)  echo "Invalid argument count" >&2
        echo "$Usage" >&2
        exit 1 ;;
esac

# Invoke the add-user binary with arguments that:
# -k /etc/skel (Copy contents of /etc/skel into new home directory)
# -d /usr2/account-name (home directory)
# -m create home directory if it doesn't exist; if it does exist,
#    it must have read, write, and search permission by primary group
# -s shell program path name
# -g defines primary group membership for new account
# -u user-id number for new account
```

```
useradd -k /etc/skel -d /usr2/$name -m -s /usr/bin/ksh -g 60001 -u $uid $name

# Install a password:

passwd $name

#end of script
```

D. Script to delete a user:

```
# @(#) deluser  Delete user script
Usage="Usage: $0 account-name"

case $# in
    1) name=$1 ;;
    *) echo "Invalid argument count" >&2
       echo "Usage" >&2
       exit 1 ;;
esac

# Invoke the delete-user binary with account-name argument:
userdel -r $name

#end of script
```

Listing 2. Scripts to update and purge password files.

A. Script to update local password files:

```
# @(#) upass.cli  Client script to update local password files

# Check that all source information files exist, exit if not:
[ ! -s /etc/passwd.org ] && { ls -l /etc/passwd.org ; exit 1; }
[ ! -s /etc/shadow.org ] && { ls -l /etc/shadow.org ; exit 1; }
[ ! -s /etc/passwd.net ] && { ls -l /etc/passwd.net ; exit 1; }
[ ! -s /etc/shadow.net ] && { ls -l /etc/shadow.net ; exit 1; }

# Reset password info
cp /etc/passwd.org /etc/passwd
cp /etc/shadow.org /etc/shadow

# Set permissions
chmod 600 /etc/passwd
chmod 600 /etc/shadow

# Append user password info
cat /usr2/passwd.net >> /etc/passwd
cat /usr2/shadow.net >> /etc/shadow

# Secure password files
chmod 444 /etc/passwd
chmod 400 /etc/shadow

#end of script ... save as upass.cli
```

B. Script to purge network password files:

```
# @(#) upass.srv  Server script to purge network password files

# If files exist, then attempt removal:
[ -s /usr2/passwd.net ] && rm /usr2/passwd.net
[ -s /usr2/shadow.net ] && rm /usr2/shadow.net

#end of script ... save as upass.srv
```

Listing 3. Program to create network password files.

```
/* Server program to create/update the network password files
 *
 *  Use superuser privilege to run this program
 *
 *  This program reads the passwd and shadow files and writes
 *  all records--after "sysop" account--to the proper network file
 *  Sets permission of network files to mode 644
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <unistd.h>
#include <time.h>

void
main()
{
    short         flag;
    unsigned char line[512];
    time_t        timer;
    FILE          *fp1, *fp2, *fp3, *fp4;

    time(&timer); /* put time since epoch in timer variable */
    printf("server password update program %s\n", ctime(&timer));

    /* Make sure invoker is superuser */

    if ((getuid() != 0) && (geteuid() != 0)) {
            puts("Must be superuser to execute");
            exit(2);
    }

    /* Open the password files */

    if ((fp1 = fopen("/etc/passwd", "r")) == NULL) {
        perror("Unable to open /etc/passwd for reading");
        exit(1);
    }

    if ((fp2 = fopen("/etc/shadow", "r")) == NULL) {
        perror("Unable to open /etc/shadow for reading");
        exit(1);
    }

    if ((fp3 = fopen("/usr2/passwd.net", "w")) == NULL) {
        perror("Unable to open /usr2/passwd.net for writing");
        exit(1);
    }
```

```
    if ((fp4 = fopen("/usr2/shadow.net", "w")) == NULL) {
        perror("Unable to open /usr2/shadow.net for writing");
        exit(1);
    }

    /* Process the passwd file: */

    flag = 0;
    while (fgets(line, 512, fp1) != NULL) {
        if (flag)
            fputs(line, fp3);
        if ((strstr(line, "sysop") != NULL) && (!flag))
            flag++;
    }

    /* Process the shadow file: */

    flag = 0;
    while (fgets(line, 512, fp2) != NULL) {
        if (flag)
            fputs(line, fp4);
        if ((strstr(line, "sysop") != NULL) && (!flag))
            flag++;
    }

    /* Set correct permissions on output files: */

    if (chmod("/usr2/passwd.net", 0644) < 0) {
        perror("Unable to set permissions on /usr/passwd.net");
        exit(2);
    }

    if (chmod("/usr2/shadow.net", 0644) < 0) {
        perror("Unable to set permissions on /usr/shadow.net");
        exit(2);
    }

    /* Close the files */

    fclose(fp1); fclose(fp2); fclose(fp3); fclose(fp4);

    /* Show long-directory listing of output files: */

    system("ls -li /usr2/*.net");

    puts("...files updated...end of program");

} /* Save executable as upass */
```

Edited by Becca Thomas / Online Editor / UnixWorld Online / beccat@wcmh.com
Software tested by John Skinner and Jesse I. Pollard.

Go To Content

Search

Last Modified: Saturday, 28-Oct-95 06:35:07 PDT