/\*

\* backup-fs.c

\* I)ruid <druid@caughq.org>

\*

\* This backup utility will push it's tar.gz through a keyauth'd

\* SSH tunnel to a remote filesystem, avoiding having to store the

\* backup locally. Will use the system fstab for backup targets or

\* a user supplied fstab file.

\*

\*/

#include <fcntl.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/stat.h>

#include <sys/types.h>

#include <time.h>

#include <unistd.h>

/\* Define your backup user and your backup host here \*/

#define BACKUP\_USER "backup"

#define BACKUP\_HOST "druid.druid.net"

#define SSH\_PORT 22

/\* my personal debuggin output \*/

#define DEBUG 1

main(int argc, char \*argv[] ) {

FILE \*f1;

int x, y;

char \*fstab;

char mountpoint[256];

char fstype[32];

char datestring[32];

char buffer[256], final[256];

char hostname[128];

char tar\_command[256];

char disk[32];

char modified\_disk[32];

/\* If an fstab file is not specified, default to /etc/fstab \*/

if( !argv[1] )

fstab = "/etc/fstab";

else

fstab = argv[1];

/\* Get Current Hostname \*/

gethostname( hostname, sizeof(hostname) );

if( DEBUG == 1 ) printf( "Hostname: [%s]\n", hostname );

/\* Get Formatted Date String \*/

sprintf( datestring, "%s", getdatestr() );

/\* Backup the fstab outside of the archive for reference \*/

sprintf( tar\_command, "/bin/cat %s | ssh -p %d -l %s %s \"cat - > incoming/%s-fstab-%s\"", fstab, SSH\_PORT, BACKUP\_USER, BACKUP\_HOST, hostname, datestring );

system( tar\_command );

/\* Parse fstab for entries \*/

f1 = fopen( fstab, "r" );

if( f1 == NULL ) {

printf( "Error opening %s\n", fstab );

exit( -1 );

}

if( DEBUG == 1 ) printf( "%s opened for reading...\n", fstab );

while( fgets( buffer, sizeof(buffer), f1 ) ) {

buffer[strlen(buffer)-1] = '\0';

if( DEBUG == 1 ) printf( "Read [%s]\n", buffer );

y = 0;

for( x = 0; x <= strlen(buffer); x++ ) {

if( buffer[x] == '\t' || buffer[x] == ' ' ) {

final[y] = ':';

y++;

x++;

while( buffer[x] == '\t' || buffer[x] == ' ' )

x++;

}

final[y] = buffer[x];

y++;

}

if( DEBUG == 1 ) printf( "Mod: [%s]\n", final);

sprintf( disk, "%s", strtok( final, ":" ) );

sprintf( mountpoint, "%s", strtok ( NULL, ":" ) );

sprintf( fstype, "%s", strtok( NULL, ":" ) );

if( DEBUG == 1 ) printf( "Disk: [%s]\n", disk );

if( DEBUG == 1 ) printf( "Mountpoint: [%s]\n", mountpoint );

if( DEBUG == 1 ) printf( "FSType: [%s]\n", fstype );

for( x = 0; x <= strlen(disk); x++ ) {

if( disk[x] == '/' ) {

if( x == 0 )

modified\_disk[x] = '-';

else

modified\_disk[x] = '.';

} else

modified\_disk[x] = disk[x];

}

if( DEBUG == 1 ) printf( "ModDisk: [%s]\n", modified\_disk );

/\* If the entry is of type ext2, we will back it up \*/

if( strcmp( fstype, "ext2" ) == 0 ) {

/\* Build tar command string \*/

sprintf( tar\_command, "/bin/tar -czl %s | ssh -p %d -l %s %s \"cat - > incoming/%s%s-%s.tar.gz\"", mountpoint, SSH\_PORT, BACKUP\_USER, BACKUP\_HOST, hostname, modified\_disk, datestring );

if( DEBUG == 1 ) printf( "Command: [%s]\n", tar\_command );

system( tar\_command );

}

}

exit( 1 );

}

/\* Return the DateString for the archive name \*/

getdatestr() {

time\_t t;

int year;

char holdme[4096];

char outfile[1024];

char month[1024];

char day[1024];

char id[1024];

/\* Get the year from the current time \*/

t = time(NULL);

sprintf( holdme, "%s", ctime( &t ) );

if( DEBUG == 1 ) printf( "Time: [%s]\n", holdme );

sprintf( id, "%s", strtok( holdme, " " ) );

sprintf( month, "%s", strtok( NULL, " " ) );

sprintf( day, "%s", strtok( NULL, " " ) );

sprintf( id, "%s", strtok( NULL, ":" ) );

sprintf( id, "%s", strtok( NULL, " " ) );

sprintf( id, "%s", strtok( NULL, "\n" ) );

year = atoi( id );

/\* End of year (y2k) Bug Fix - If the log is for Dec 31, then it will be

the next year when the process above finds the year (if executed at/near

midnight), therefore we go back a year \*/

if( !strcmp( month, "Dec" ) && !strcmp( day, "31" ) )

year--;

/\* Convert the Month to a number value (so things list incrementally in

filesystem list \*/

if( strcmp( month, "Jan" ) == 0 )

sprintf( month, "01" );

if( strcmp( month, "Feb" ) == 0 )

sprintf( month, "02" );

if( strcmp( month, "Mar" ) == 0 )

sprintf( month, "03" );

if( strcmp( month, "Apr" ) == 0 )

sprintf( month, "04" );

if( strcmp( month, "May" ) == 0 )

sprintf( month, "05" );

if( strcmp( month, "Jun" ) == 0 )

sprintf( month, "06" );

if( strcmp( month, "Jul" ) == 0 )

sprintf( month, "07" );

if( strcmp( month, "Aug" ) == 0 )

sprintf( month, "08" );

if( strcmp( month, "Sep" ) == 0 )

sprintf( month, "09" );

if( strcmp( month, "Oct" ) == 0 )

sprintf( month, "10" );

if( strcmp( month, "Nov" ) == 0 )

sprintf( month, "11" );

if( strcmp( month, "Dec" ) == 0 )

sprintf( month, "12" );

/\* Pad the day with 0's so it's at least 2 char's \*/

if( strlen( day ) < 2 ) {

sprintf( id, "0%s", day );

sprintf( day, "%s", id );

}

/\* Archive the logs \*/

sprintf( outfile, "%d.%s.%s", year, month, day );

if( DEBUG == 1 ) printf( "DateString: [%s]\n", outfile );

return( outfile );

}