Primer Certamen
Tiempo 15:40 hrs - 17:10 hrs.
Responder un problema por página

1.- (30 puntos Shell Script) Usted se entera que no es seguro tener scripts bash (Bourne-Again shell) con su bit set-user-ID fijado. Se propone entonces buscar en el sistema por tales archivos. Escriba un script bash llamado wrongPerm.sh. Para un directorio dado como parámetro, éste lista por pantalla todos los script bash que tengan fijado el bit set-user-ID (para cambiar el usuario efectivo). Un ejemplo de su invocación es:
$ ./wrongPerm.sh . /* se omite salida */
Hints: Revise el comando ***file*** para identificar script bash. Por ejemplo:
$ file myScript
myScript: Bourne-Again shell script, ASCII text executable

Revise la opción –u del comando ***test*** para reconocer archivos con bit set-user-ID fijado. También puede considerar la opción -perm de comando ***find***.

```
#!/bin/bash
find $1 -perm -4000 | while read setuidFile
do
  file $setuidFile | grep "Bourne-Again" | awk '{print $1}'
done
```

Otra forma es:

```
#!/bin/bash
dir=$1
for i in $dir/*
do
   if test -d "$i"
   then
      $0 $i
   else
      b=`file $i | awk '{print $3 }'`
      if [ -u "$i" -a "$b" = "Bourne-Again" ]
      then
         echo $i
      fi
   fi
done
```

2.- (35 puntos) Manejo de Procesos. Esta pregunta explora el efecto en proceso hijo y padre de una alarma configurada en proceso padre. Desarrolle un programa que configure el envío futuro de una señal de alarma y luego genere un proceso hijo. Su programa debe mostrar por pantalla mensajes que permitan concluir a qué procesos llegó la alarma. ¿Cuál es el resultado que usted espera para la salida de su programa?

```
#include <signal.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>

pid_t parentPID;

static void sig_alarm(int signo) {
   if (signo == SIGALRM) {
     if (parentPID == getpid())
       printf("Parent received ALARM\n");
     else
       printf("Child received ALARM\n");
```

```
   }
   return;
}

int main(void) {
  pid_t pid;

  parentPID = getpid();
  if (signal(SIGALRM, sig_alarm) == SIG_ERR)
      exit(-1);
  alarm(2);
  if ( (pid = fork()) < 0)
    exit(-1);
  else {
    pause();
    sleep(2);
    if (pid != 0) {   /* parent */
       kill(pid,9);
       waitpid(pid, NULL, 0);
    }
  }
}
```

3.- (35 puntos) En esta pregunta se explora el tiempo que una hebra está en estado “ready” (o listo) en espera por hacer uso de la CPU (éste es el estado running).
Haga un programa que corra dos hebras. Una de ellas calcula el tiempo máximo que la hebra estuvo esperando por usar la CPU y la otra mostrará ese valor por pantalla. Como idea un Sansano sugiere usar la función gettimeofday en un loop de N iteraciones, con N ingresado como argumento del programa. La primera hebra calcula el valor máximo de la diferencia entre dos invocaciones a esta función, este máximo debería corresponder al tiempo de la hebra fuera de la CPU. La segunda hebra se limita a esperar por el cambio del valor máximo y lo muestra por pantalla.
Idea:
Hebra 1: |-|-|-|-|-|-|-|-|-|-|-|….....|-|-|-|-|-|-|-|-|-|-|-|-|-.....
← running → ← ready →
Hebra 2: ^ muestra máximo

int gettimeofday(struct timeval *tv, struct timezone tz); /* tz puede ser NULL.*/
struct timeval {
time_t tv_sec; /* seconds */
suseconds_t tv_usec; /* microseconds */
};
No cuestione el método; si prefiere, puede usar otro siempre que haya interacción entre ambas hebras.

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <sys/time.h>
#include <signal.h>

pthread_mutex_t maxLock = PTHREAD_MUTEX_INITIALIZER;
pthread_cond_t maxChange = PTHREAD_COND_INITIALIZER;

int max=0;

void * printMax(void * arg)
{
  while (1) {
    pthread_mutex_lock(&maxLock);
    pthread_cond_wait(&maxChange, &maxLock);
    printf("Max time in ready state so far: %i [us]\n",max);
    pthread_mutex_unlock(&maxLock);
```

```
  }
}

int main(int argc, char * argv[]) {
  pthread_t tid;
  int err, i;
  struct timeval ti, tf;
  int time;

  int N = atoi(argv[1]);
  err = pthread_create(&tid, NULL, printMax, NULL);
  if (err != 0){
    printf("can't create thread \n");
    exit(0);
  }
  gettimeofday(&ti, NULL);
  for (i=N; i>0 ; i--) {
    gettimeofday(&tf, NULL);
    time=(tf.tv_sec-ti.tv_sec)*1000000+(tf.tv_usec-ti.tv_usec);
    if (time > max) {
      pthread_mutex_lock(&maxLock);
      max=time;
      pthread_mutex_unlock(&maxLock);
      pthread_cond_signal(&maxChange);
    }
    ti=tf;
  }
  exit(0);
}
```