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

1.- (30 puntos Shell Script) Cree el script shell noUsanCuenta.sh el cual permite determinar el porcentaje de cuentas con acceso a una máquina pero nunca se han conectado a éste. El script retorna el número total de cuentas, número de aquellas que nunca se han conectado a la máquina y el porcentaje que éstas representas. Su ejecución arroja algo del tipo:
$ noUsanCuenta.sh
Total: 550
Nunca login: 55
Porcentaje: 10%
Ayuda: La salida de finger para una cuenta que nunca se ha usado en la máquina es:

```
agustin.gonzalez@Aragorn:~$ finger wolfgang.freund
Login: wolfgang.freund              Name: Wolfgang Freund
Directory: /home/profesores/wolfgang.freund    Shell: /bin/bash
Never logged in.
No mail.
No Plan.
```

```
#! /bin/bash
IFS=":"
numCounts=0
numNever=0
getent passwd >> /tmp/noUsanCuenta$$
while read userName rest
do
  numCounts=$(($numCounts + 1))
  if finger $userName |grep "Never logged in" >>/dev/null
  then
      numNever=$(($numNever +1))
  fi
done < /tmp/noUsanCuenta$$
echo "Total: $numCounts"
echo "Nunca login: $numNever"
echo "Porcentaje: `expr $numNever \* 100 / $numCounts`%"
rm /tmp/noUsanCuenta$$
```

2.- (35 puntos) Se desea explorar el tiempo que un proceso se encuentra en estado corriendo en la CPU y el tiempo es estado ready (esperando por volver a hacer uso de la CPU).
Para esto se sugiere hacer un programa que corra un loop M veces. M es ingresado como argumento del programa. En cada iteración guarda en un arreglo de tamaño M la diferencia Δt entre dos llamados consecutivos para obtener la hora. Cuando el arreglo se llena, el programa muestra un histograma de los valores Δt.

Ayuda: Considere la función hist de octave:
Function File: **hist** *(y, nbins) :* With one vector input argument, *y*, plot a histogram of the values with *nbins*.
La función en Linux:

```
int gettimeofday(struct timeval *tv, struct timezone *tz);
The  functions  gettimeofday()  can get the time as well as a timezone.  The tv
argument is a  struct  timeval:
           struct timeval {
```

```
               time_t      tv_sec;     /* seconds */
               suseconds_t tv_usec;    /* microseconds */
           };
If either tv or tz is NULL, the corresponding structure is not set.

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

int main(int argc, char * argv[]) {
    pid_t pid;
    int pfd[2], i, status, *delta;
    FILE * sd;
    struct timeval tvi, tvf;
    int M = atoi(argv[1]);

    delta = malloc(M*sizeof(int));
    gettimeofday(&tvi, NULL);
    for (i=0; i<M; i++) {
       gettimeofday(&tvf, NULL);
       delta[i]= (tvf.tv_sec-tvi.tv_sec)*1000000+(tvf.tv_usec-tvi.tv_usec);
       tvi=tvf;
    }
    if (pipe(pfd) < 0) {
        perror("pipe");
        exit(1);
    }
    if ((pid = fork()) < 0) {
        perror("fork");
        exit(1);
    }
    if (pid == 0) {
        int junk;
        dup2(pfd[0], 0);
        close(pfd[1]);  /* close the write end of the pipe */
        junk = open("/dev/null", O_WRONLY);
        dup2(junk, 1);  /* throw away any message sent to screen*/
        dup2(junk, 2);  /* throw away any error message */
        execlp("octave", "octave", "-i", (char*)NULL);
        perror("exec");
        exit(-1);
    }
    close(pfd[0]);
    sd = fdopen(pfd[1], "w");  /* to use fprintf instead of just write */
    fprintf(sd, "delta= [");
    for ( i= 0; i < M; i++)
      fprintf(sd, "%d ", delta[i]);
    fprintf(sd, "];\n");
    fprintf(sd, "hist(delta);\n");
    fflush(sd);
    sleep(10);
    fprintf(sd, "\n exit\n"); fflush(sd);
```

```
    waitpid(pid, &status, 0);
    fclose(sd);
    exit(0);
}
```

3.- (35 puntos) Se desea generar un flujo de números aleatorios a un periodo definido en cada instante por el usuario. Los números aleatorios son escritos en un archivo y su periodo es de n segundos entre datos, con n entero y en segundos y según lo especificado por el usuario vía teclado. Para atender este requerimiento se propone crear dos hebras. Una espera por un nuevo valor de periodo por parte del usuario. La otra escribe datos aleatorios a intervalos regulares según el periodo definido por el usuario.
Nota: por simplicidad use sleep(n) para el tiempo entre datos generados, con n en segundos.

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

int periodo=1;
pthread_mutex_t mylock = PTHREAD_MUTEX_INITIALIZER;

void * lecturaDePeriodo(void * arg)
{
   int p;
   do {
       printf("Ingrese nuevo periodo: ");
       scanf("%i", &p);
       pthread_mutex_lock(&mylock);
       periodo=p;
       pthread_mutex_unlock(&mylock);
   } while (periodo > -1);
   exit(0);
}

int main(int argc, char * argv[]) {
  pthread_t tid;
  FILE * file = fopen(argv[1], "w");
  int err, p;

  err = pthread_create(&tid, NULL, lecturaDePeriodo, NULL);
  if (err != 0){
    printf("can't create thread \n");
    exit(0);
  }
  while (1) {
    fprintf(file,"%i\n",rand()); fflush(file);
    pthread_mutex_lock(&mylock);
    if (periodo > 0) p=periodo;
    else break;
    pthread_mutex_unlock(&mylock);
    sleep(p);
  }
  close(file);
}
```