CERTAMEN #1 INF-250

Wenceslao Palma <wenceslao.palma@ucv.cl>

1. Responda las sgtes preguntas. Justifique cada una de sus respuestas.

   (a) (5 ptos.) En el algoritmo de scheduling Round-Robin los procesos nuevos se ubican al final de la cola y no al inicio de ésta. Por qué?
   R.: si los procesos nuevos se ubican al inicio de la cola pueden perjudicar a los procesos llegados con anterioridad. De este modo es posible que cada vez que un proceso va a pasar al estado *running* llegue un proceso nuevo lo cual provocará inanición.

   (b) (5 ptos.) Es posible que un proceso en estado *ready* pase directamente al estado *blocked*?
   R.: no, para que un proceso pase al estado *blocked* debe hacerlo desde el estado *running*.

   (c) (6 ptos.) Los comandos **$wc -l < temp** y **$wc -l temp** generan la misma salida, cuál es la diferencia?
   R.: **< temp** es interpretado por el shell, de este modo **wc** no considera el archivo **temp** como un argumento sino como su entrada estándar. En cambio en **$wc -l temp** el archivo **temp** es considerado un argumento del comando **wc**.

2. (20 ptos.) Considere los siguientes datos:

Proceso	Llegada	Tiempo de Servicio
A	0	3
B	1	2
C	3	2
D	6	5
E	9	6

Muestre como los algoritmos FCFS y RR(q=1). Realice una tabla con los valores de Tiempo de Finalización, $T_{retorno}$ y $T_{retorno}/T_{servicio}$. Para cada algoritmo realice el gráfico correspondiente. Con todo lo realizado comente.

R.:

FCFS (9 ptos.)

```
    0...3 4 5 6 7 8 9 10 11 12.....18
A    XXX
B        X X
C            X X
D                X X X  X  X
E                            XXXXXX
```

	A	B	C	D	E	Media
Tfinalizacion	3	5	7	12	18	
Tretorno	3	4	4	6	9	
Tr/Ts	1	2	2	1.2	1.5	1.54

RR (q=1) (9 ptos.)

```
     0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
A     X   X     X
B       X   X
C             X   X
D                   X X    X     X     X
E                       X     X     X     X  X  X
```

	A	B	C	D	E	Media
Tfinalizacion	6	4	7	15	18	
Tretorno	6	3	4	9	9	
Tr/Ts	2	1.5	1.5	1.6	1.5	1.62

comentario (2 ptos)

3. (a) (14 ptos.) Escriba un programa en C que usando 2 threads acepte datos desde la entrada estándar (thread 1) y cada cierto tiempo tome los datos ingresados y los almacene en un archivo llamado buffer.txt (thread 2).

```
#include <stdio.h>
#include <pthread.h>
#include <unistd.h>
#include <fcntl.h>
#include <string.h>


void *readBuf(void *);
void *writeBuf(void *);

char buffer[200]="";

void main ()
{
   pthread_t thread1,thread2;
   int id1=1,id2=2;

   printf("creando thread 1\n");
   pthread_create(&thread1, NULL, readBuf, (void *)id1);

   printf("creando thread 2\n");
   pthread_create(&thread2, NULL, writeBuf, (void *)id2);

   pthread_exit(NULL);
}


void *readBuf(void *ThreadID){

  printf("ingrese texto:\n");
  while (1)
    fgets(buffer,200,stdin);
}

void *writeBuf(void *ThreadID){
  int fd;


  while (1){
     if ((fd = open("buffer.txt",O_WRONLY|O_CREAT, 0666)) == -1){

        fprintf(stderr, "problemas al abrir el archivo.......\n");
```

```
    }
     sleep(3);
     write(fd,buffer,strlen(buffer));
     close(fd);
  }
}
```

(b) (10 ptos.) Utilizando los datos contenidos en /proc, escriba un script usando bash y awk que muestre todos los procesos que se encuentran en estado RUNNING.

```
#!/bin/bash

for pid in `ls /proc/ | grep ^[0-9]`
do
  if [ -e /proc/$pid/status ]
  then
       state=$(grep State /proc/$pid/status | awk '{print $2}')
       if [ "$state" == "R" ]
       then
            echo $pid
       fi
  fi
done
```