SISTEMAS OPERATIVOS - CUESTIONES
28 de Junio de 2016

Nombre _____ DNI_____

Apellidos _____ Grupo _____

Cuestión 1. (0,75 puntos - 0,25p cada una) En un sistema con 32 Kbytes de memoria RAM, las direcciones virtuales son de 16 bits, de los cuales los 4 bits más altos referencian la página de memoria virtual y los 12 bits más bajos son el offset dentro de la página.
a) ¿Cuántas páginas hay y de qué tamaño son? ¿Qué tamaño de memoria virtual maneja el procesador?
b) Si en este momento se están ejecutando dos procesos y el contenido de la memoria física es el que se muestra a continuación, indicar el contenido actual de la tabla de páginas (TP) del proceso1 y del proceso2.
c) ¿Se producirá un fallo de página si arrancamos la ejecución de un tercer proceso? Razonar la respuesta.

Memoria Principal

Marco	
0	Pág. 0 - Proceso1
1	Pág. 1 - Proceso 1
2	
3	Pág. 0 – Proceso 2
4	TMP
5	Pág. 2 – Proceso 1
6	Pág. 1 – Proceso 2
7	

TP Proceso1		**TP Proceso2**	
Pág. 0		Pág. 0	
Pág. 1		Pág. 1	
Pág. 2		Pág. 2	
Pág. 3		Pág. 3	

Cuestión 2. (1 punto) Hilos: contesta a las preguntas sobre el siguiente código, razonando tu respuesta.:

```
void *trabajo(void *arg);
volatile int variable_global = 0;

pthread_t tid[3]; //tid es el identificador de hilo (threadID)

int main(void)
{
   int i = 0;
   while(i < 3)
   {
      pthread_create(&(tid[i]), NULL, trabajo, NULL);
      printf("\n Hilo creado!\n");
      i++;
   }
   //Espera a que todos los hilos finalicen
   (void) pthread_join(NULL);
   exit (0);
}
```

```
void *trabajo(void *arg)
{
   int i = 0;
   pthread_t id = pthread_self();

   while(i<8)
   {
        i++;
        variable_global++;
         if(pthread_equal(id,tid[0])){
                sleep(1);
                i++;
         }
         else
            if(pthread_equal(id,tid[1])){
                sleep(2);
                i=9;
            }
             else
                sleep(3);
   }
}
```

a) **(0,5 p)** ¿Cuántos hilos se crean? ¿Cuántas veces se ejecuta cada hilo (while)?
b) **(0,25 p)** ¿Cuál es el valor final de variable_global?
c) **(0,25 p)** ¿Qué cambiaría de las respuestas anteriores si no hubiéramos escrito los pthread_join?

Cuestión 3. (1 punto) Contestar a las siguientes preguntas sobre diferentes formas de planificar la ejecución de este conjunto de tareas.

a) **(0,33 p)** ¿Cuál será la última tarea en ejecutarse si se aplica SJF no expropiativo? ¿Qué productividad se consigue?
b) **(0,33 p)** ¿Cuál sera el tiempo total de ejecución con Round Robin y q=3? ¿Qué productividad se consigue?
c) **(0,33 p)** ¿Qué tarea se estará ejecutando en el ciclo 12 si se aplica un algoritmo expropiativo basado en prioridad que tome como tarea más prioritaria la de menor número en el campo Prioridad?

Tarea	Llegada	Prioridad	CPU	E/S	CPU
T1	0	3	3	2	2
T2	1	2	5		
T3	2	2	4	1	
T4	0	4	6	1	
T5	4	1	3	2	

Nota: marcar el tiempo de CPU coloreando y el de E/S rayando

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
T1																									
T2																									
T3																									
T4																									
T5																									

SJF no expropiativo

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
T1																									
T2																									
T3																									
T4																									
T5																									

Round Robin q=3

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
T1																									
T2																									
T3																									
T4																									
T5																									

PRIO expropiativo

Fórmula: Productividad = (nº de tareas)/(nº de ciclos)

Cuestión 4. (1 punto) Un programador ha escrito el siguiente código para resolver un problema tipo lector/escritor. Contestar a las siguientes preguntas:

a) (0,25 p) ¿Cuántos lectores y/o escritores puede haber realizando trabajo efectivo simultáneamente?
b) (0,25 p) Si la salida por pantalla ha sido 5, 7, 7, 13 ¿cuál ha sido la secuencia de ejecución?
(Ejemplo: L-L-E)
c) (0,5 p) Modificar el código para que solamente pueda haber un escritor o hasta 3 lectores simultáneos.

```
int dato = 5;

void *Lector(void *arg);
void *Escritor(void *arg);

pthread_mutex_t cerrojo;

main(int argc, char *argv[])  {
    pthread_t th1, th2, th3, th4;
    pthread_mutex_init (&cerrojo,NULL);

    pthread_create(&th1, NULL, Lector, NULL);
    pthread_create(&th2, NULL, Escritor, NULL);
    //Suponer que se crean tantos hilos Lector y Escritor
    //como sean necesarios

   pthread_join(NULL);
   pthread_mutex_destroy(&cerrojo);
   exit(0);
}
```

```
void* Lector(void *arg) {
   while(1){
      pthread_mutex_lock(&cerrojo);
      printf("%d\n", dato);
      pthread_mutex_unlock(&cerrojo);
  }
}

void* Escritor(void *arg){
  while(1){
     pthread_mutex_lock(&cerrojo);
        dato = dato + 2;
     pthread_mutex_unlock(&cerrojo);
   }
}
```

Cuestión 5. (1 punto) Dada la siguiente secuencia de referencias a direcciones de memoria virtual generadas por un sólo programa en un sistema con paginación pura:

`0x3A0, 0x4B8, 0x210, 0x320, 0x40C, 0x6F8, 0x504, 0x122, 0x22C, 0x43A, 0x380, 0x6D2, 0x502`

a) (0,25 p) Obtén la cadena de referencias (secuencia de números de página generadas por el programa), suponiendo un tamaño de página de 256 Bytes..

b) (0,75 p) Determina razonadamente el número de fallos de página usando como estrategia de sustitución el algoritmo del reloj, suponiendo que hay cuatro marcos de página disponibles para el programa y que están inicialmente vacíos.

REF.	0x3A0	0x4B8	0x210	0x320	0x40C	0x6F8	0x504	0x122	0x22C	0x43A	0x380	0x6D2	0x502
#pag													
m0													
m1													
m2													
m3													

Cuestión 6. (1,25 punto) Un proceso en UNIX ejecuta el siguiente código:

```
int main(void)
{
  pid_t  PIDhijo;
  pid_t soy, papa;
  int child_status;
  int var= 7;

  soy=getpid();
  archivo1=open("prueba.txt",O_RDWR);

  PIDhijo = fork();

  if (PIDhijo == 0)
  {
    int j;
    char buf[1] = {0};
    char c;
    int numB=0;

    for (j = 0; j < 14; j++) var=var+1;
    while((c=read(archivo1,buf, 1))!=EOF)
            numB = numB+1;
    exit(0);
  }
  else
  {
    int j;
    int numB=0;
    char c;
    for (j = 0; j < 6; j++)  var=var+1;

   while((c=getc(archivo1))!=EOF)
          numB = numB+1;

    wait(&child_status);

    close(archivo1);

    exit(0);
  }
  exit (0);
}
```

Nota: *Prueba.txt contiene "Contesta las siguientes preguntas:\n"*

Contesta las siguientes preguntas:

a) (0,5 p)¿Qué valor tiene la variable **var** antes de que se ejecute el fork()? ¿Qué valor tiene la variable **var** del padre y la del hijo inmediatamente después de ejecutarse el fork()? ¿Qué valor tiene la variable **var** del padre y del hijo cuando termine de ejecutarse el padre?

b) (0,25 p) Rellenar las siguientes tablas (tabla de descripción de archivos, tabla de archivos intermedia y tabla de nodos-v) si se supone que se han realizado en orden las siguientes ejecuciones 10 vueltas del while del padre, 5 vueltas del while del hijo y 2 vueltas del while del padre. Cuánto valdría **c**.

c) (0,25 p)¿Qué valor tiene la variable **numB** del padre y del hijo después de terminar de ejecutarse el padre?

d) (0,25 p) ¿Es posible que alguno de los dos procesos lea correctamente el archivo?

TDDA padre

0	230
1	564
2	28
3	12
4	

TDDA hijo

0	230
1	564
2	28
3	
4	

TFA (intermedia)

	Pos L/E	**num nodo-i**	**permisos**	**cont. refs.**
...				
12		57		
13				
14				
...				

Tabla nodos-v

Nodo-v	**Contador**
...	
Info nodo-57	
...	

Problema 1 (2 puntos): Se ha diseñado un sistema operativo que organiza la información en el disco duro mediante i-nodos, estos están organizados con 2 índices directos y 1 índice indirecto para bloques de 32B. El sistema operativo gestiona un disco duro que en un momento dado tiene la siguiente información:

i-nodo	2	4	7	8	9	10	11	16	
enlaces	4	3	2	1	1	1	1	2	
Tipo	D						F		
Directo		2	4	5	10	15	9		
Directo	null	null	null	6	12	8	null	null	
Indirecto	null	null	null	null	null	7	null	null	

Tabla de nodos-i

1		2		3		4		7
.	2	.	4	.	16	.		18
..	2	..	2	..		..		19
home	4	estudiante	16	Examen.odt	8			
usr	7	tmp.txt	11	Notas.txt	9			
				Solucion.odt	10			

Lista de bloques relevantes (los bloques que no aparecen aquí contienen datos o están vacíos)

a. (0,25 ptos) Rellena el mapa de bits de bloques de datos (hasta el bloque 20), donde 0 es libre y 1 ocupado, el primer índice se corresponde con el bloque 0.
b. (0,25 ptos) Rellena los datos que faltan en la tabla de i-nodos y en la lista de bloques relevantes
c. (0,5 ptos) Dibuje el árbol de directorios
d. (0,5 ptos) Si se ejecuta el siguiente código, indiqué qué se verá modificado en las dos tablas y por qué (añadir las columnas a la tabla si fuera necesario).
   /home$ mkdir compatida
   /home$ cp estudiante/Notas.txt compartida/Notas.txt
e. (0,5 ptos) Considere el siguiente código, indique qué llamada al sistema modificará seguro las tablas anteriores, describa cómo se quedaría la Tabla de i-nodos y por qué (añadir las columnas a la tabla si fuera necesario).

…

```
char buf[32] = ”31 carac. mas el salto de carro\n”;

int fd = open(“/home/tmp.txt”, O_RDWR);

lseek(fd, 32, SEEK_SET);

write(fd, buf,32);
```

Problema 2 (2 puntos): La Facultad de Informática ha decidido implantar baños unisex. Para poder realizar este paso es necesario tener un programa informático que asegure que mientras el baño está siendo utilizado por mujeres sólo podrán entrar al baño mujeres y exactamente igual con los hombres. Para lo que se utilizará un display (es decir, una variable) a la entrada del baño que indicará si está vacío (display = 0), ocupado por mujeres (display = 1) u ocupado por hombres (display = 2).

Para comprobar que el programa funciona se crea un programa principal que genera diferentes hilos de tipo mujer y tipo hombre con el siguiente código:

int mujer()
{
  mujer_desea_entrar();
  actividad_baño_m();
  mujer_sale();
  return 1;
}
	int hombre()
	{
	  hombre_desea_entrar();
	  actividad_baño_h();
	  hombre_sale();
	  return 1;
	}

a) (1,5 ptos) Implementar las funciones *mujer_desea_entrar()*, *mujer_sale()*, *hombre_desea_entrar()* y *hombre_sale()* utilizando mutex y variables condicionales.

Para poder puntuar de una manera más justa el código contestar a las siguientes preguntas:

1. Qué tipo de problema tipo representa este ejercicio.
2. Cómo se tiene que inicializar el mutex y las variables condicionales en main.
3. Cuantas variables se tienen que proteger y por qué.

b) (0,5 ptos) Modificar el código anterior para permitir sólo 3 mujeres o 5 hombres a la vez en el baño (podéis utilizar semáforos si consideráis que facilita la programación de esta parte)