

junio2019.pdf



TuTorMentor



Sistemas Operativos



3º Grado en Ingeniería Informática



Facultad de Informática
Universidad Complutense de Madrid

Maths
informática



CURSOS INTENSIVOS PARA EXÁMENES DE

**CONVOCATORIA ORDINARIA
Y EXTRAORDINARIA**

CURSOS INTENSIVOS PARA EXÁMENES DE CONVOCATORIA ORDINARIA Y EXTRAORDINARIA



SISTEMAS OPERATIVOS - CUESTIONES
18 de junio de 2019

Nombre _____ DNI _____
Apellidos _____ Grupo _____

Cuestión 1. (1 punto) Comente en qué función (inicialización, finalización, apertura, lectura, etc.) y por qué hemos utilizado las siguientes llamadas dentro del módulo del kernel chardev estudiado en clase:

a) `module_put (THIS_MODULE);` / `try_module_get (THIS_MODULE);`
(++) en open (ui) -- en release (ui)

Para prevenir el borrado de un módulo que está en uso
Incrementa / Decrementa el contador interno del kernel por cada módulo
y es distinto a 0 no puede eliminarse
\$ lsmod para consultar el valor

b) `unsigned long copy_to_user(void __user * to, const void * from, unsigned long n);`

para hacer una copia segura de los datos de usuario en el espacio
del kernel, porque el driver puede no tener acceso
en read (lectura)

c) `unsigned long copy_from_user(void * to, const void __user * from, unsigned long n);`

para hacer una copia segura de los datos desde espacio de kernel
en write (escritura)

d) Indique qué comando se usaría en el terminal para crear un dispositivo tipo carácter llamado exam_dev al que se le ha asignado un major/minor de 250/1 con permisos rw-r-r-

\$ mknod /dev/exam-dev c 250 1 -m 644



Academia especializada en estudios
de la Facultad de Informática

Maths
informática

WUOLAH

Escaneado con CamScanner

Cuestión 2. (1 punto)

Las siguientes tablas representan la FAT y el contenido del directorio raíz de un pequeño sistema de ficheros con un tamaño de bloque de 1KB (las entradas vacías estarían marcadas como libres). Se pide:

- Complete la tabla que representa el directorio
- Indique cómo se modificarán la tabla FAT y el directorio cuando se ejecute el comando make para compilar el proyecto, creando así un ejecutable llamado prog de 3640 Bytes.

FAT

0	4
1	8
2	5
3	6
4	1
5	EOF
6	7
7	EOF

8	EOF
9	EOF
10	11
11	12
12	13
13	EOF
14	
15	

Directorio Raíz

Nombre	Tipo	Tamaño (B)	Bloque
makefile	F	1300	2
main.c	F	2500	3
funcs.c	F	3400	0
funcs.h	F	60	9
prog	F	3640	10

Cuestión 3. (1 punto) En un sistema monoprocesador llegan a procesarse cuatro tareas con los siguientes patrones de ejecución:

Trabajo	Llegada	CPU	E/S	CPU	E/S
P1	0	4	2	3	
P2	1	3	3	4	2
P3	2	4	2	3	

Simule la ejecución de las tareas utilizando los diagramas de Gantt de abajo. Calcule el porcentaje de uso de CPU, así como el tiempo de espera de cada una de las tareas cuando se aplican los algoritmos de planificación SJF (no expropiativo) y RR-3 (Round Robin con cuanto 3). ¿Qué planificador tiene menor tiempo de espera promedio? Al rellenar los diagramas, usar el siguiente convenio para representar los estados de las tareas:

- En ejecución: marcar con "X"
- Bloqueado por E/S: marcar con "O"
- Listo para ejecutar: marcar con "--"

SJF	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	
P1	X	X	X	X	-	-						X	X	X								
P2					X	X	X	-	-	-								X	X	X	X	
P3								X	X	X	X	-	-		X	X	X					

Porcentaje de uso de CPU: _____ %
 Tiempo de espera de P1: _____ unidades
 Tiempo de espera de P2: _____ unidades
 Tiempo de espera de P3: _____ unidades

CURSOS INTENSIVOS PARA EXÁMENES DE
**CONVOCATORIA ORDINARIA
Y EXTRAORDINARIA**

Asegúrate un aprobado

Clases online en directo

Resolución de exámenes de las
últimas convocatorias

Gana confianza y agilidad en la resolución
de ejercicios de exámenes

Campus virtual con
material de ayuda

Tutorías y resolución
de dudas

RR	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20
P1	X	X	X																		
P2				X	X	/	-	-	-									X	-	-	
P3							/	/	/										X	X	X

Porcentaje de uso de CPU: _____ %
 Tiempo de espera de P1: _____ unidades
 Tiempo de espera de P2: _____ unidades
 Tiempo de espera de P3: _____ unidades

Cuestión 4. (1 punto) La función `system()` de la biblioteca estándar de "C" permite crear un nuevo proceso shell hijo que invoca el comando que se pasa como parámetro a la función, y devuelve el código de salida (*status*) del shell.

```
int system(const char *command);
```

El proceso de usuario que invoca `system()` se queda bloqueado hasta que el comando finalice.

Implemente la función `system()` empleando llamadas al sistema. (**Nota:** Para ejecutar un comando se puede invocar al shell de la siguiente forma: `/bin/bash -c '<comando>'`. Ejemplo: `/bin/bash -c 'ls -l'`).

Cuestión 5. (1 punto) El siguiente código pretende resolver el problema de la sección crítica para dos hilos H0 y H1.

<pre>// Variable compartida por H0 y H1. int interested[2] = {0,0} H0(){ while(1) { interested[0]=1; while(interested[1]==1) { interested[0]=0; while(interested[1]==1) { } interested[0]=1; } SECCION_CRITICA_H0(); interested[0]=0; } }</pre>	<pre>H1(){ while(1) { interested[1]=1; while(interested[0]==1) { interested[1]=0; while(interested[0]==1) { } interested[1]=1; } SECCION_CRITICA_H1(); interested[1]=0; } }</pre>
---	--

CURSOS INTENSIVOS PARA EXÁMENES DE CONVOCATORIA ORDINARIA Y EXTRAORDINARIA

A. Enumere y describa qué criterios debe cumplir cualquier mecanismo que solucione el problema de la sección crítica.

B. Indique justificadamente cuáles de los anteriores criterios se cumplen o no en la solución propuesta.

Cuestión 6. (1 punto) Un sistema utiliza gestión de memoria virtual con páginas de 4 KiB, direcciones de memoria de 64 bits y memoria principal de 16 GiB. Se quiere ejecutar en dicho sistema el siguiente programa:

```
#include ...

#define MAX(X, Y) (((X) > (Y)) ? (X) : (Y))

/* La función lee un vector del
 * fichero cuya ruta se pasa como
 * parametro. Reserva memoria con
 * malloc y devuelve un puntero
 * al vector y el número de elementos.
 */
int* lee_vector( char* fichero, int* n_elementos )
{
    ... // malloc ...
}

/* Devuelve el mayor elemento del vector
 * en el rango [start_idx:end_idx]*/
int mayor( int* vector, int start_idx, int end_idx )
{
    ...
}

int elmayor;

int main(int argc, char *argv[]) {
    int n_elem;
    int* vector;
    pid_t pid;
    int max1=0, max2=0;

    /* A */
    vector=lee_vector(argv[1],&n_elem);

    /* B */
    if ((pid=fork())==0) {
        /* Proceso hijo */
        max1=mayor(vector,0,n_elem/2);
        exit(0);
    } else if (pid>0) {
        /* Proceso padre */
        max2=mayor(vector,n_elem/2,n_elem);
    } else {
        fprintf(stderr,"Error en fork()\n");
        exit(2);
    }

    /* C */
    /* Esperar al proceso hijo */
    while(wait(NULL)!=pid) {} ;
    elmayor = MAX(max1,max2);
    printf("El mayor elemento del vector es %d\n", elmayor);

    free(vector);
    /* D */
    return (0);
}
```

Considerando que:

- Hay suficientes marcos de página libres en el sistema.
- El texto (código) del programa tras la compilación ocupa 3 KiB.
- Las bibliotecas son dinámicas y ocupan 9 KiB.
- El contenido inicial de la pila al lanzar a ejecución el programa ocupa 512 bytes.
- Un entero en C ocupa 4 bytes.
- El fichero que se pasa como argumento al programa contiene 1024 enteros.



Academia especializada en estudios de la Facultad de Informática



Identifique qué regiones lógicas (y su correspondiente tamaño en páginas) constituyen la imagen de memoria del proceso o procesos que se crean al lanzar a ejecución dicho programa en los puntos marcados en el código como A, B, C y D.

punto A

codigo	1
pila	1

punto B

codigo	1
heap	1
lib dinamic	2
pila	1

punto C

codigo	1
heap	1
lib dinamic	2
pila	1

codigo	1
heap	1
lib dinamic	2
pila	1

punto D

se libera el heap



SISTEMAS OPERATIVOS - PROBLEMAS

18 de junio de 2019

Nombre _____ DNI _____

Apellidos _____ Grupo _____

Problema 1. (2 puntos)

Supóngase un sistema de ficheros tipo UNIX con las siguientes características:

- i-nodos de 48 bytes (para descripción y atributos del fichero) más 8 punteros directos, un puntero indirecto simple y un puntero indirecto doble.
- Tamaño de los punteros a bloques y a i-nodos: 32 bits.
- Tamaño de bloques de 16 KiB

1. (0.25pts) Con los datos indicados ¿Cuál es el tamaño del fichero más grande que se puede almacenar en este sistema de ficheros?
2. (0.25pts) ¿Cuántos i-nodos caben en un bloque de i-nodos?
3. (0.25pts) Suponiendo que los directorios tienen entradas con nombres de tamaño fijo de 116 caracteres. ¿Cuántos bloques de datos ocuparía un directorio con 1024 entradas?
4. (0.5pts) Si en este sistema se ejecutasen las siguientes llamadas al sistema y suponiendo que buff es un buffer de 24 KiB con un contenido distinto de cero:

```
fd=open("mifichero", O_RDWR|O_CREATE|O_TRUNC);
lseek(fd, 1024*1024, SEEK_SET);
write(fd, buff, 24*1024);
close(fd);
```

5. ¿Qué tamaño tendría el fichero y cuántos bloques de datos y bloques de indirección quedarían ocupados? Dibujar la estructura final del fichero "mifichero" indicando el contenido de las entradas relevantes de los bloques de indirección.
6. (0.25pts) Si usamos FAT, ¿Qué tamaño tendría el fichero y cuántos bloques ocuparía?
7. (0.25pts) ¿Cuántos i-nodos y bloques de datos nuevos se ocuparán al crearse un enlace físico que apunte a "mifichero"?
8. (0.25pts) ¿Cuántos i-nodos y bloques de datos nuevos se ocuparán al crearse un enlace simbólico que apunte a "mifichero"?

① $32/8 = 4B$; $16KB/4B = 4KB = 4096B$

tamaño por organización

$$8 * 2^{14} + 2^{12} * 2^{14} + 2^{12} * 2^{14} = 274GB$$

tamaño por puntero L/E

$$2^{32}B = 4.2GB$$

tamaño por

$$2^{32} * 2^{14} = 70TB$$

② Tamaño inodo = $48B + 10pts * 4 = 88$

$$16384 / 88 = 186 \text{ inodos}$$

③ $\frac{1024 * 116}{16384} = 7.25 \text{ bloques}$

④ 66 bloques de los que contienen datos son los 2 últimos.

⑤ si fuera FAT ocupa los 66 bloques.

⑥ el enlace físico se crea inodos nuevos aumenta el nº de enlaces en el inodo correspondiente a "mifichero"

⑦ crea un inodo nuevo que contiene la ruta hacia "mifichero"

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.