

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

Midterm

IIC 2333 — Sistemas Operativos y Redes
17 de Mayo de 2024

Duración: 2 horas. Cada parte se responde en una **hoja aparte**

Parte 1: Procesos y Syscalls [16p]

Respecto a la materia de Procesos, responda de manera breve y precisa las siguientes preguntas.
Corrección: Las oraciones que son falsas otorgar **0.5 pts** si puso que es falsa y **0.5 pts** por la explicación.

1.1 **[6p]** Responda V o F, Justificando las falsas.

(a) Un proceso esta compuesto por el código, datos, stack y los recursos que utiliza (Ej:Memoria).

- **Respuesta:** Verdadero

(b) Un proceso padre comparte el mismo PCB que sus hijos.

- **Respuesta:** Falso, siempre es un PCB por cada proceso.

(c) El cambio de procesos se conoce como Context Switch.

- **Respuesta:** Verdadero

(d) La Syscall `exec()` corre un archivo ejecutable, luego retoma la ejecución del código original.

- **Respuesta:** Falso. `exec()` reemplaza el código de quién lo llamó, por lo tanto no se retoma la ejecución del original.

(e) Después de ejecutar `fork()`, el proceso nuevo puede modificar el valor de las variables que ha obtenido de su creador sin afectar las variables del creador.

- **Respuesta:** Verdadero

(f) El costo de un cambio de contexto en un procesador se debe principalmente a la necesidad de reemplazar los datos *Process Control Block*.

- **Respuesta:** Verdadero

1.2 **[2p]** Explique que es un procezo Zombie. **Respuesta:** Proceso que terminó y su padre aún no captura el estado de salida.

1.3 Respecto a la materia de syscalls y procesos, responde las siguientes preguntas a partir del siguiente código.

```
1
2  void signal_handler(int sig) {
3      printf("Terminando proceso con señal %d\n", sig);
4      exit(sig);
5  }
6
7
8  int main(int argc, char *argv[]) {
9      signal(SIGTERM, signal_handler); // SIGTERM = 15
```

```
10      pid_t pid;
11      int status;
12
13      pid = fork();
14      if (pid == 0) {
15          while(1);
16      };
17
18      kill(pid, SIGTERM);
19      wait(&status);
20      print("%d\n", WEXITSTATUS(status));
21
22      for(int i = 0; i < 5; i++) {
23          pid = fork();
24          if (pid != 0) {
25              wait(&status)
26          };
27          execl("/bin/ls", "/bin/ls", NULL);
28      }
29  }
```

(a) **[2p]** ¿Cuántos procesos son creados? No considere el proceso principal.

Respuesta:

- **1 pts** Hay dos procesos creados.
- **1 pts** Uno en la línea 13 y otreo en la línea 23. No se realizan todas las iteraciones dado que tanto el padre cómo el hijo hacen `exec`.

(b) **[1p]** Considere que la función `WEXITSTATUS(status)`, se utiliza para extraer el código de salida del estado capturado en `status`. ¿Qué debería imprimir la línea 20?

Respuesta: 15.

(c) **[3p]** A partir del código, identifique y explique para que sirven las syscalls y las señales involucradas.

- **0.5 pts** fork: Genera un duplicado del proceso actual, es decir, es una copia del proceso padre. La función crea un proceso hijo a partir del proceso padre y devuelve el PID (Identificador de proceso) del hijo.
- **0.5 pts** waitpid: Se utiliza para que el proceso padre espere a que sus procesos hijos terminen su ejecución. Esta syscall se encuentra en el segundo bucle `for` y espera a que cada uno de los procesos hijos termine para continuar con la ejecución del proceso padre.
- **0.5 pts** exec (execl): Reemplaza el proceso por otro y comúnmente utilizada por los programas para ejecutar otros programas en el sistema operativo. Además, incluye varias funciones con diferentes nombres y argumentos, en el código se encuentra execl().
- **0.5 pts** kill: Para enviar una señal a un proceso.
- **0.5 pts** exit: Para terminar un proceso y entregar código de salida.
- **0.5 pts** SIGTERM: Señal utilizada para solicitar al programa que termine.

(d) **[2p]** Suponga reeplazamos todos los '`SIGTERM`' del código por '`SIGKILL`', cuyo valor númerico es 9 ¿Qué cambios se verían en la consola al momento de ejecutar el programa? Explique la razón de estos cambios.

Respuesta: No se mostraría el print de la línea 3 **(+1p)**, debido a que la señal SIGKILL no es manejable **(+1p)**.

Parte 2: Threads y Sincronización [17p]

2.1 **[6p]** Respecto a la materia de Threads y Sincronización, responda V o F, justificando las falsas de manera breve. **Corrección:** Las oraciones que son falsas otorgar **0.5 pts** si puso que es falsa y **0.5 pts** por la explicación.

(a) Thread es la unidad basica de CPU.

- **Respuesta:** Verdadero

(b) Los Threads no comparten Thread ID, el Program Counter y Registros.

- **Respuesta:** Verdadero

(c) La función `yield()` permite entregar el turno a otros threads.

- **Respuesta:** Verdadero

(d) Un thread en el espacio del usuario tienen mejor escalabilidad.

- **Respuesta:** Verdadero, en el espacio del usuario hay escalabilidad

(e) Un sistema multicore no puede implementar multitasking, pero un sistema single-core sí.

- **Respuesta:** Falso, multicore y singlecore puede tener multitasking

(f) Cuando implemento los threads en el espacio del usuario, las syscalls no bloquean los procesos.

- **Respuesta:** Falso

2.2 **[2p]** Describa que es una condición de carrera y que hacer para evitarlas.

(a) **Respuesta:** Cuando dos threads/hilos acceden a una variable compartida al mismo tiempo o quieren acceder a la misma sección de código **(+1 pts)**. Para evitarlas se tiene que:

a) Deshabilitar las interrupciones.
   (a) Una solución con variable compartida *lock*
   (b) Una solución con Variable compartida *turn* que determina quién puede entrar
b) Por hardware: Instrucciones atómicas que pueden ser test_and_set() y compare_and_swap()
c) Semáforos
d) Mutex Lock

Importante: Basta que nombre y explique alguna de las soluciones anteriores (+1 pts). También es válido que nombre varias soluciones.

2.3 Las syscall, como servicios del sistema operativo, ejecutan en kernel space. Un proceso de usuario, por otro lado, no puede ejecutar arbitrariamente instrucciones en kernel space, ni escribir en memoria del kernel, ni cambiar arbitrariamente el modo de protección.

(a) **[2p]** ¿Cómo consigue un proceso de usuario que el código de una syscall se ejecute en modo kernel?
Respuesta:. Se usan:

- **1 pts** bibliotecas de usuarios que preparan el llamado a la syscall
- **1 pts** interrupciones para pasar el control al kernel

(b) **[2p]** ¿Cómo consigue un proceso de usuario entregarle argumentos a una syscall si no puede escribir en memoria del kernel?

Respuesta: Los argumentos se dejan en una posición de la memoria del usuario **(+1p)** (técnicamente en el stack pero no es necesario ser tan específico), y después de la interrupción, el kernel copia los datos de la memoria del usuario a la memoria del kernel **(+1p)**.

En caso de no mencionar que se copian los datos, o bien se deja ver que el código del usuario hace la copia antes de llamar la interrupción **(1p)**.

2.4 La próxima vez que vaya a la sala de ayuda y tenga que solicitar ayuda a uno de los ayudantes, piense que está ejecutando un protocolo de exclusión mutua entre múltiples *threads*. Suponga que existe solo un ayudante (1 CPU), y `n` estudiantes (*threads*) que desean recibir ayuda.

Para `n` estudiantes, se tiene un arreglo compartido `necesitaAyuda[0:n-1]`, donde `necesitaAyuda[i]` indica si el estudiante `i` esta pidiendo ayuda. Se cuenta además con la variable compartida `turnoActual` el cual indica a que estudiante le toca ser atendido.

Por último, se tiene la función `RecibirAyuda(i)` la cual se utiliza para representar que el estudiante `i` esta siendo atendido, esta función solo debe estar siendo ejecutada por a lo más un *thread*. También, se cuenta con la función `ContinuarTarea(i)` que representa la acción del estudiante que continua realizando su tarea, esta función puede puede ser ejecutada por múltiples threads en paralelo.

Se propone el siguiente código para modelar el protocolo de exclusión mutua (para simplificar consideramos solamente 3 estudiantes):

```
// variables compartidas
bool necesitaAyuda[3] = {false, false, false};
int turnoActual = 0;
```

```
/** Codigo para thread i **/
while (true) {
  necesitaAyuda[i] = true;
  turnoActual = (i + 1) % 3;
  while ((necesitaAyuda[(i+1) % 3] || necesitaAyuda[(i+2) % 3]) && turnoActual != i);
  /** Inicio Sección Crítica **/
  RecibirAyuda(i);
  necesitaAyuda[i] = false;
  /** Final Sección Crítica **/
  ContinuarTarea(i);
}
```

A partir del código anterior responda la siguientes preguntas:

(a) **[3p]** Argumente si en la(s) sección(es) críticas se cumplen las propiedades de solución al problema de la **sección crítica**.

Respuesta:

- **1 pts** Exclusión mutua **no se cumple**, ya que cada thread le entrega el turno al siguiente con la línea `turnoActual=(i+1) %3`. Con esto, es posible que un thread `i` le entregue el turno al thread `(i+1) %3` y antes que este último salga de la SC thread `(i-1) %3` le entregué el turno al thread `i`.

- Espera acotada **no se cumple**, un thread `i` puede ceder el turno al thread `(i + 1) % 3`, pero antes de que logre entrar a la SC, el thread `(i-1) %3` cede el turno al thread `i`, luego el `(i+1) %3` no entrará, esperarando infinitamente.

- Progreso **no se cumple**, si un thread `i` e `(i+1) %3` quieren entrar a la SC y el `turnoActual` es igual a `(i-1) %3`, y este thread se encuentra en `ContinuarTarea(i)`, entonces los otros dos se quedarán esperando.

(b) **[2p]** Modifique el código de tal manera que se cumplan todas las propiedades antes mencionadas. Su código debe incluir las funciones `RecibirAyuda(i)` y `ContinuarTarea(i)` y una primitiva de software. Indique las variables compartidas que se requieran.

Respuesta: Varias respuestas. Por ejemplo, podemos utilizar *locks* o semáforos binarios. Una solución con *locks* sería. Considerar que es incorrecto incluir `ContinuarTarea(i)` dentro de la sección crítica.

Variable compartida: `lock`
Código thread `i`:

```
%/** Codigo para thread i **/
%while (true) {
%    lock.aquire();
%    /** Inicio Sección Crítica **/
%    RecibirAyuda(i);
%    /** Final Sección Crítica **/
%    lock.release();
%    ContinuarTarea(i);
%}
```

Parte 3: Scheduling [14p]

3.1 **[5p]** Respecto a la materia de scheduling, responda V o F, justificando las falsas de manera breve. **Corrección:** Las oraciones que son falsas otorgar **0.5 pts** si puso que es falsa y **0.5 pts** por la explicación.

(a) El scheduler *Shortest Job First (SJF)* **no** puede provocar *starvation*.

- **Respuesta:** Falso. El scheduler SJF puede provocar *starvation* debido a que nunca dará turno a los procesos que poseen rafagas de CPU-burst largos.

(b) La acción de *Swap* es sacar la memoria del proceso de la memoria principal y llevarlos al espacio del kernel.

- **Respuesta:** Falso, se llevan al espacio de swap

(c) El scheduler *Multi-Level Feedback Queue Scheduling (MLFQ)* permite interactividad entre los procesos de manera priorativa.

- **Respuesta:** Verdadero

(d) Los procesos con menor CPU-Burst siempre se ejecutarán primero en un scheduler FIFO.

- **Respuesta:** Falso, en FIFO es el primero que llega será ejecutado. La definición corresponde a SFJ.

(e) En el espacio de usuario se decide que tarea elegir en un scheduler.

- **Respuesta:** Falso, lo decide el sistema operativo. Además, ocurre en el kernel.

3.2 **[2p]** Mencione dos diferencias entre *Round Robin Scheduler* y *Last In, First Out (LIFO) Scheduler*.

- **Respuesta:** Por cada mención otorgar **1 pts**.

Round Robin Scheduler	**LIFO Scheduler**
Proceso se ejecuta por un quantum de tiempo	Proceso más reciente se ejecuta primero
Si no termina su ejecución, se mueve al final de la cola	Último proceso en entrar es el primero en salir
Equitativo, cada proceso recibe un tiempo de CPU	No garantiza equidad, procesos más recientes pueden tener prioridad
Menor tiempo de espera para procesos pequeños	Puede causar inanición de procesos más antiguos

Cuadro 1: Diferencias entre RR y LIFO scheduler.

3.3 Para evaluar el rendimiento de los schedulers *Shortest Job First (SJF)* y *First In, First Out (FIFO)* en la ejecución de procesos con una sola *CPU-Burst*. Se provee la siguiente información.

Proceso	Tiempo inicial(Llegada)	CPU-Burst
P1	0	15
P2	0	5
P3	2	10

3.3.1) **[4p]** A partir de la tabla anterior, complete la siguiente tabla. **Por cada número correcto en la tabla dar 0.25 pts.**

	FIFO		SJF	
Proceso	Response Time	Turnaround Time	Response Time	Turnaround Time
P1	0	14	15	29
P2	14	19	0	4
P3	17	27	3	12
Promedio	10,333	20	6	15

Cuadro 2: Tabla de tiempos de respuesta y tiempo de retorno para los algoritmos FIFO y SJF. **Importante:** En caso de que los valores no sean los correctos por una diferencia de un +-1, en los resultados presentados pero se respeta que en FIFO se ingresa P1 - P2 - P3 y en SJF es P2 - P3 - P1. Solo descontar -0.3 pts.

3.3.2) Justifique las siguientes afirmaciones. **Importante: Si no hay datos respaldados por la tabla, poner 0 pts.**

a) **[1.5 p]** El response time promedio del scheduler *Shortest Job First (SJF)* es mayor que el scheduler *First In, First Out (FIFO)*.
- **1.5 pts** Según la tabla es menor.

b) **[1.5 p]** El turnaround time promedio del scheduler *Shortest Job First (SJF)* es mayor que el scheduler *First In, First Out (FIFO)*.
- **1.5 pts** Según la tabla es menor.

Parte 4: Memoria y Archivos [13p]

4.1 **[6p]** Memoria

4.1.1) **[2p]** Respecto a la materia de memoria, responda Verdadero o Falso en las siguientes preguntas, justifique las falsas.
Corrección: Las oraciones que son falsas otorgar **0.5 pts** si puso que es falsa y **0.5 pts** por la explicación.

(a) El tamaño de la tabla de página invertida crece por el tamaño de direcciones virtuales.
- **Respuesta:** Falso. La tabla de página invertida crece por el tamaño de la memoria física.

(b) Los algoritmos de reemplazo de páginas buscan minimizar la cantidad de Page Fault y aumentar el hit rate.
- **Respuesta:** Verdadero

4.1.2) **[1p]** Explique la diferencia entre un marco (frame) y una página (segmento).
- **1 pts** Frame es de la memoria física
- **1 pts** Página es de la memoria virtual
- Otra respuesta posible: Un frame es lo que tiene tu computador, mientras que la página virtual es lo que el proceso çree"que es la memoria.

4.1.3) **[1p]** Una estructura de memoria presenta la siguiente distribución.

Tabla PCBs	Tamaño de entrada	256 Bytes
	Cantidad de entradas	32
	Tamaño en memoria fisica	8KB
Tabla Invertida de Página	Tamaño de entrada	3 Bytes
	Cantidad de entradas	65536
	Tamaño en memoria fisica	192KB
Frame Bitmap	Tamaño en memoria fisica	8KB
Frames	Tamaño en memoria fisica	2GB

Cuadro 3: Distribución memoria física.

Si en vez de una tabla invertida de páginas, se utiliza una tabla de páginas por cada proceso que es agrega a su PCB, es decir, cada PCB ahora sería los 256B de información más la tabla de página. Considere que cada entrada de la tabla de páginas es de 4B y que se utilizan 13 bits para el número de página (VPN) ¿Qué tamaño tendría la memoria física?. No es necesario simplificar.

- **Respuesta:** Cada entrada de la tabla de páginas 4B y hay 2^{13} entradas en cada tabla de páginas, por lo tanto, una tabla de páginas es de $3 * 2^{13}$
  Una entrada de la PCB usa $4 * 2^{13} + 2^8$
  La tabla de PCBs usa $(4 * 2^{13} + 2^8) * 2^5 = 2^{20} + 2^{13} = 1MB + 8KB$
  Antes teniamos 8KB + 192 KB + 8 KB + 2 GB, ahora tenemos 1 MB + 8 KB + 8 KB + 2 GB, aca usamos mas espacio

4.1.4) **[2p]** Para las siguientes preguntas utilice una memoria física de 64 frames, direcciones físicas de 25 bits y una memoria virtual de 32 GB:

a) Determine el tamaño de la memoria física, la cantidad de páginas, el tamaño de los frames y el largo de una dirección virtual (en bits):
   - **0.25 pts:** 32MB
   - **0.25 pts:** 2^{16} paginas
   - **0.25 pts:** 512KB
   - **0.25 pts:** 35 bits

b) Indique cual(es) de las siguientes direcciones físicas pueden ser la traducción de la siguiente dirección virtual 0x1E94FA8B1:
   - **0.5 pts Respuesta:** Las direcciones correctas son 0x8A7A8B1 y 0xD8FA8B1 **0.5 pts**.
   - **0.5 pts Explicación:**
     - (son 9 dígitos para dir virtual)-¿pero primer bit no es parte de la dirección
     - 7 digitos fisica pero primeros 3 no son parte de la dirección, siempre 0
     - frames 512KB $->$ 19 bits $->$ 5 digitos offset hexa primer bit no es offset

4.2 **[7p]** Pregunta Archivos

4.2.1) **[2p]** En un sistema de archivos determinado se está intentando crear un nuevo archivo vacío en un directorio existente, por lo que es necesario modificar tres bloques: el bitmap, el bloque de directorio y el bloque índice del archivo. Durante este proceso hay una caída del sistema y no se alcanzan a actualizar todos los bloques. Asuma que no se ejecutan herramientas de verificación de consistencia como `FSCK` (Unix) o `chkdsk` (Windows). Dadas las siguientes posibles consecuencias (considere que estas se pueden presentar inmediatamente o en un futuro a causa de esta caída):

(a) No hay ni habrá inconsistencias (parecerá como si la operación simplemente no se hubiera hecho)
(b) Hay un *data leak* (similar a un *memory leak*, pero para disco)
(c) Con distintos *file paths* se podría llegar al mismo bloque índice
(d) Hay punteros que apuntan a cualquier cosa
(e) Varios problemas de los antes mencionados

Cuál de estas es la que corresponde si **solamente** se alcanzaron a actualizar los siguientes bloques:

a) Sólo bitmap

   - **0.5 pts Respuesta** (b)
   - **0.5 pts. Por la explicación** Se marca en el bitmap un bloque como ocupado, pero no hay nada que esté apuntando a él. Si en intentos sucesivos ocurre lo mismo, se irán marcando cada vez más bloques como ocupados cuando en realidad no lo están.

b) Sólo bloque índice de archivo

- **0.5 pts Respuesta** (a)
- **0.5 pts. Por la explicación** Se escribe un bloque índice que no está siendo apuntado y no está marcado como ocupado, por lo que todo lo que se escribió puede ser ignorado.

4.2.2) **[2p]** Respecto a la materia de Archivos, responda Verdadero o Falso en las siguientes preguntas, justifique las falsas:

(a) Un archivo es la unidad mínima de almacenamiento para el usuario.
Respuesta: Verdadero

(b) En los hard links no hay distinción entre el archivo original y el enlace al archivo, solo tiene dos nombres para el mismo archivo, por lo tanto eliminar solo uno de los nombres no eliminara el otro.
Respuesta: Verdadero

4.2.3) **[3p]** Indique 3 caracteristicas de un disco en estado sólido (SSD).

Respuesta: 1 punto por cada caracteristicas

- Sin seek time ni latencia rotacional
- Más rápidos
- Más costoso por MB
- Menor consumo eléctrico
- Celdas poseen cantidad limitada de reescrituras