SO: 1er Parcial

Galileo Cappella

2c2022

Disclaimer y Notas

En el exámen me saqué un 95 con [25, 25, 20, 25].
Las correcciones están marcadas con un bloque gris.
Cualquier respuesta es posible que diferentes profesores la corrijan diferente, recomiendo que te fijes cómo la explicarías vos si no te convence.
¡Mucha suerte!

Question 1: Sincronización y concurrencia (25 puntos)

Se cuenta con N procesos y tres funciones que estos procesos deben ejecutar. Estas funciones son:

- `primera_f()`
- `segunda_f()`
- `tercera_f()`

Cada función debe ser ejecutada por un solo proceso a la vez, pero para que algún proceso pueda ejecutar la función `segunda_f()`, todos los N procesos deben haber ejecutado la función `primera_f()`, y para que puedan ejecutar la función `tercera_f()`, todos los N procesos deben haber ejecutado la función `segunda_f()`. Finalmente cuando todos los N procesos hayan terminado de ejecutar la función `tercera_f()`, deben volver a iniciar la ejecución de las tres funciones: `primera_f()`, `segunda_f()`, `tercera_f()`, según las restricciones y órden planteados, y este ciclo se repetiría de forma indefinida.

Se pide que cree un programa (pseudocódigo basado en C), que permita la ejecución de los N procesos, según las reglas mencionadas. Suponga que los N procesos ya han sido creados previamente.

Solution:

```
//S: Variables globales entre procesos
int N, //U: Cantidad de procesos, la asumo como dada
    done = 0; //U: Cantidad de procesos que terminaron este paso
semaphore b1(0), b2(0),
          mtx(1);

//S: Codigo
while (1) {
  mtx.wait(); //A: Espero a mi turno
  primera_f();
  if (++done == N) b1.signal(N); //A: Si soy el ultimo, dejo pasar al resto
  mtx.signal(1); //A: Libero para el siguiente turno
  b1.wait();

  mtx.wait();
  if (--done == 0) b2.signal(N); //A: Soy el ultimo dejo pasar
  mtx.signal(1);
  b2.wait();

  //A: Similar a la primera_f
  mtx.wait();
  segunda_f();
  if(++done == N) b1.singal(N);
  mtx.signal(1);
  b1.wait();

  mtx.wait();
  if (--done == 0) b2.signal(N); //A: Soy el ultimo dejo pasar
  mtx.signal(1);
  b2.wait();

  //A: Similar a las otras dos
  mtx.wait();
  tercera_f();
  if(++done == N) b1.singal(N);
  mtx.signal(1);
  b1.wait();

  mtx.wait();
  if (--done == 0) b2-signal(N);
  mtx.signal(1);
  b2.wait();
}
```

Hay seis secciones críticas, entre los `mtx.wait()` y `mtx.signal()`. Siempre todos los procesos liberan los recursos.

Similarmente, todos los procesos pasan las barreras, y siempre esperan a pasar b2 a que todos hayan pasado b1, ya pasar b1 a que todos hayan corrido la función.

Question 2: Scheduling (25 puntos)

Considere que el Sistema Operativo utiliza un algoritmo de scheduling multinivel feedback queue compuesto de tres colas.
Utiliza una política con desalojo. En la siguiente tabla se muestran las ráfagas de CPU de cuatro (4) procesos. La cola 1 funciona con un quantum de 19ms, la cola 2 funciona con un quantum de 21ms, y la tercera cola usa FCFS. La cola 1 tiene mayor prioridad que la 2, y la 2 que la 3. Todos los procesos comienzan en la cola 1.

Procesos	CPU Burst (ms)	Tiempo de llegada (ms)
P1	57	0
P2	19	1
P3	71	2
P4	27	3

Se pide:

a) Dibuje el diagrama de Gantt correspondiente.

b) Calcule el turnaround promedio.

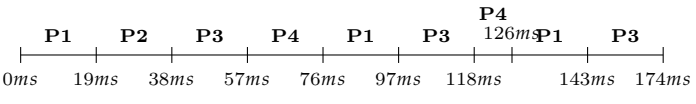
c) Calcule el waiting time promedio.

d) ¿Cuántas veces es interrumpido cada proceso desde que inicia hasta que termina su ejecución?

e) ¿En qué cola termina su ejecución cada proceso?

Solution:

a)



b y c)

Proceso	Llegada	CPU Burst	Finalización	Turnaround	Waiting
P1	0ms	57ms	143ms	143ms - 0ms = 143ms	143 - 57ms = 86ms
P2	1ms	19ms	38ms	38ms - 1ms = 37ms	37ms - 19ms = 18ms
P3	2ms	71ms	174ms	174ms - 2ms = 172ms	174 - 71ms = 103ms
P4	3ms	27ms	126ms	126ms - 3ms = 123ms	123 - 27ms = 96ms
Promedio	-	-	-	118.75ms	75.75ms

d)

Proceso	Interrupciones
P1	2
P2	0
P3	2
P4	1

e)

Proceso	Última cola
P1	3
P2	1
P3	3
P4	2

Question 3: Comunicación Inter-procesos (25 puntos)

Dados 2 procesos, un proceso padre y otro proceso hijo. Complete el código y las funciones `HABLAR_AL_PADRE()`, `HABLAR_AL_HIJO()`, `ESPERAR_AL_PADRE()`, `ESPERAR_AL_HIJO()`, y `MANEJADOR()` para que el proceso padre e hijo se comuniquen via pipes solamente (usando `pipefd1` y `pipefd2`). Esta comunicación debe permitir obtener la salida mostrada. Los procesos deben terminar después de 10 segundos usando la system call `alarm()`.
Ayuda: la system call `alarm(N)` hace que el proceso se envíe a sí mismo una señal `SIGALARM` después de N segundos.
Nota: Solo deben completar el código en las partes marcadas con `TODO`. Cada proceso debe cerrar los pipes que no utiliza.

La salida esperada es la siguiente:

```
El padre le envía un mensaje al hijo!
El hijo recibe un mensaje desde el padre!
El hijo le envía un mensaje al padre!
El padre recibe un mensaje desde el hijo!
El padre le envía un mensaje al hijo!
...
...
Alarma:  el tiempo terminó
```

Código a completar (atención a todas las partes marcadas con `TODO`, son 8):

```
int pipefd1[2], pipefd2[2]; //globales

void MANEJADOR(int signum) {
  print("Alarma: el tiempo termino");
  // TODO1
}

void PIPES(void) {
  if (pipe(pipefd1) < 0 || pipe(pipefd2) < 0) {
    perror("pipe");
    exit(EXIT_FAILURE);
  }
}

void HABLAR_AL_PADRE(void) {
  // TODO2
  printf("El hijo envia un mensaje al padre!\n");
}

void ESPERAR_AL_PADRE(void) {
  // TODO3
  printf("El hijo recibe un mensaje desde el padre!\n");
}

void HABLAR_AL_HIJO(void) {
  // TODO4
  printf("El padre envia un mensaje al hijo!\n");
}

void ESPERAR_AL_HIJO(void) {
  // TODO5
  printf("El padre recibe un mensaje desde el hijo!\n");
}

int main(int argc, char* argv[]) {
  PIPES();
  pid_t pid;
  pid = fork();
  // TODO6. Manejo de senales.
  alarm(TODO7);
  if (TODO8) {
    while (1) {
```

```
      sleep(rand()%2 + 1);
      HABLAR_AL_HIJO();
      ESPERAR_AL_HIJO();
    }
  } else {
    while (1) {
      sleep(rand()%2 + 1);
      ESPERAR_AL_PADRE();
      HABLAR_AL_PADRE();
    }
  }
  return 0;
}
```

Solution:

```
//TODO1:
exit(EXIT_SUCCESS); //NOTA: No hace falta cerrar pipes en exit

//TODO2:
int msg = 1; //NOTA: Da igual que dice el mensaje
if (write(pipefd2[1], &msg, sizeof(msg)) < 0) { //A: Error escribiendo
  perror("HABLAR_AL_PADRE\n");
  exit(EXIT_FAILURE);
}

//TODO3:
int msg;
read(pipefd1[0], &msg, sizeof(msg)); //A: Espera hasta leer algo

//TODO4:
int msg = 1; //NOTA: Da igual que dice el mensaje
if (write(pipefd1[1], &msg, sizeof(msg)) < 0) { //A: Error escribiendo
  perror("HABLAR_AL_HIJO\n");
  exit(EXIT_FAILURE);
}

//TODO5:
int msg;
read(pipefd2[0], &msg, sizeof(msg)); //A: Espera hasta leer algo

//TODO6:
if (pid == 1) {
  perror("fork\n");
  exit(EXIT_FAILURE);
} else if (pid != 0) { //A: Soy el padre
  close(pipefd1[0]);
  close(pipefd2[1]);
} else { //A: Soy el hijo
  close(pipefd1[1]);
  close(pipefd2[0]);
}
signal(sigalarm, MANEJADOR);

//TODO7:
alarm(10);

//TODO8:
if (pid != 0) { //A: Soy el padre //NOTA: pid == -1 ya fue revisado
```

Note:-

Ojo: Según donde caigan los cambios de contexto puede que los mensajes salgan en orden distinto a del enunciado.

Question 4: Gestión de memoria (25 puntos)

Considere la siguiente secuencia de referencias a páginas: 1, 1, 2, 2, 3, 4, 5, 6, 1, 2, 4

a) Realice el seguimiento de cada uno de los algoritmos de reemplazo listados abajo, considerando que el sistema cuenta con 4 *frames* (todos ellos inicialmente libres). Además, indique el *hit-rate* para cada algoritmo.

I) LRU.

II) Segunda oportunidad.

b) Es posible disminuir el hit-rate del algoritmo que obtuvo el valor mayor, cambiando el orden de solicitud de las páginas. Muestre un ejemplo en el cual el hit-rate del algoritmo ganador ahora sea el más bajo. Use las misma solicitudes de páginas.

Solution:

a)

	LRU			Second Chance		
Ref	Frames	Orden (# de frame)	hit?	Frames	Orden (# de frame)	hit?
1	[1, _, _, _]	1, 2, 3, 0	X	[1, _, _, _]	1, 2, 3, 0	X
1	[1, _, _, _]	1, 2, 3, 0	✓	[1, _, _, _]	1, 2, 3, $\bar{0}$	✓
2	[1, 2, _, _]	2, 3, 0, 1	X	[1, 2, _, _]	2, 3, $\bar{0}$, 1	X
2	[1, 2, _, _]	2, 3, 0, 1	✓	[1, 2, _, _]	2, 3, $\bar{0}$, $\bar{1}$	✓
3	[1, 2, 3, _]	3, 0, 1, 2	X	[1, 2, 3, _]	3, $\bar{0}$, $\bar{1}$, 2	X
4	[1, 2, 3, 4]	0, 1, 2, 3	X	[1, 2, 3, 4]	$\bar{0}$, $\bar{1}$, 2, 3	X
5	[5, 2, 3, 4]	1, 2, 3, 0	X	[1, 2, 5, 4]	3, 0, 1, 2	X
6	[5, 6, 3, 4]	2, 3, 0, 1	X	[1, 2, 5, 6]	0, 1, 2, 3	X
1	[5, 6, 1, 4]	3, 0, 1, 2	X	[1, 2, 5, 6]	$\bar{0}$, 1, 2, 3	✓
2	[5, 6, 1, 2]	0, 1, 2, 3	X	[1, 2, 5, 6]	$\bar{0}$, $\bar{1}$, 2, 3	✓
4	[4, 6, 1, 2]	1, 2, 3, 0	X	[1, 2, 4, 6]	3, 0, 1, 2	X
Hit rate		$\frac{2}{11}$			$\frac{4}{11}$	

b) Si es posible, con la secuencia 4, 4, 1, 2, 3, 5, 1, 1, 2, 2, 6

	LRU			Second Chance		
Ref	Frames	Orden (# de frame)	hit?	Frames	Orden (# de frame)	hit?
4	[4, _, _, _]	1, 2, 3, 0	X	[4, _, _, _]	1, 2, 3, 0	X
4	[4, _, _, _]	1, 2, 3, 0	✓	[4, _, _, _]	1, 2, 3, $\bar{0}$	✓
1	[4, 1, _, _]	2, 3, 0, 1	X	[4, 1, _, _]	2, 3, $\bar{0}$, 1	X
2	[4, 1, 2, _]	3, 0, 1, 2	X	[4, 1, 2, _]	3, $\bar{0}$, 1, 2	X
3	[4, 1, 2, 3]	0, 1, 2, 3	X	[4, 1, 2, 3]	$\bar{0}$, 1, 2, 3	X
5	[5, 1, 2, 3]	1, 2, 3, 0	X	[4, 5, 2, 3]	2, 3, 0, 1	X
1	[5, 1, 2, 3]	2, 3, 0, 1	✓	[4, 5, 1, 3]	3, 0, 1, 2	X
1	[5, 1, 2, 3]	2, 3, 0, 1	✓	[4, 5, 1, 3]	3, 0, 1, $\bar{2}$	✓
2	[5, 1, 2, 3]	3, 0, 1, 2	✓	[4, 5, 1, 2]	0, 1, $\bar{2}$, 3	X
2	[5, 1, 2, 3]	3, 0, 1, 2	✓	[4, 5, 1, 2]	0, 1, $\bar{2}$, $\bar{3}$	✓
6	[5, 1, 2, 6]	0, 1, 2, 3	X	[6, 5, 1, 2]	1, $\bar{2}$, $\bar{3}$, 0	X
Hit rate		$\frac{5}{11}$			$\frac{3}{11}$	