**SISTEMES OPERATIUS**

**Grau en Enginyeria Informàtica, Grau en Disseny i Desenvolupament de Videojocs**

**2n Curs**

**14 de maig de 2020.**

**Segona prova:** Temps 1:30 hores.

**JUSTIFICA TOTES LES RESPOSTES**

**NOM:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ No. UdG: \_\_\_\_\_\_\_\_\_\_\_\_\_**

1. **Processos (4,0 punts)**

Es té el següent codi incomplet a on hi falten com a mínim les crides de creació de pipes. Els executables “aaa”, “bbb” i “ccc” són filtres.

|  |  |
| --- | --- |
| 1 | int fda[2], fdb[2], pid1, pid2, pid3, estat; |
| 2 |  |
| 3 | int main() |
| 4 | { |
| 5 | pid1 = fork(); |
| 6 | if (pid1 == 0) |
| 7 | { |
| 8 | close(1); |
| 9 | dup(fda[1]); |
| 10 | close(0); |
| 11 | dup(fdb[0]); |
| 12 | execl("./aaa", "aaa", (char \*)0); |
| 13 | exit(1); |
| 14 | } |
| 15 | else |
| 16 | { |
| 17 | pid2 = fork(); |
| 18 | if (pid2 == 0) |
| 19 | { |
| 20 | close(1); |
| 21 | dup(fdb[1]); |
| 22 | close(0); |
| 23 | open("./ddd",O\_RDONLY); |
| 24 | execl("./bbb", "bbb", (char \*)0); |
| 25 | signal(SIGINT, SIG\_DFL); |
| 26 | exit(1); |
| 27 | } |
| 28 | else |
| 29 | { |
| 30 | signal(SIGINT, SIG\_IGN); |
| 31 | pid3 = fork(); |
| 32 | if (pid3 == 0) |
| 33 | { |
| 34 | close(0); |
| 35 | dup(fda[0]); |
| 36 | close(1); |
| 37 | open("./eee",O\_WRONLY|O\_TRUNC|O\_CREAT); |
| 38 | execl("./ccc", "ccc", (char \*)0); |
| 39 | exit(1); |
| 40 | } |
| 41 | else |
| 42 | { |
| 43 | while(pid3!=wait(&estat)); |
| 44 | exit(0); |
| 45 | } |
| 46 | } |
| 47 | } |
| 48 | } |

1. (0,75 punts) Fes una línia de comandes que es comporti de la forma més semblant possible a com es comporta el programa anterior.

./bbb < ./ddd | ./aaa | ./ccc > ./eee

Els signals també es poden programar des del shell, però no es pot fer que només afecti a alguns dels processos connectats per pipes. Mireu man trap o man bash

Per ignorar el SIGINT:

trap ”” INT

Per deixar per defecte SIGINT:

trap - INT

1. (0,75 punts) Fes un diagrama de la jerarquia de processos que es crea amb el codi anterior. Indica per a cada procés quin programa executa.

pid1 pid2 pid3

1. (0,50 punts) De les línies següents, indica amb una X en quines es podrien crear les pipe **fda** i f**db**. La creació es faria a l’inici de la línia.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Línies** | **fda** | **fdb** |  | **Línies** | **fda** | **fdb** |
| **5** | **X** | **X** |  | **20** |  |  |
| **6** |  |  |  | **30** |  |  |
| **8** |  |  |  | **32** |  |  |
| **17** |  |  |  | **34** |  |  |
| **18** |  |  |  | **43** |  |  |

1. (0,50 punts) Indica quins valors té la taula de signals pel signal SIGINT, en el moment **de finalitzar l’execució del procés que executa les línies** de la taula següent.

|  |  |  |
| --- | --- | --- |
| **Linies** | **SIGINT** | **Justificació** |
| **8** | default | No es programa ni pel pare i per ell |
| **20** | default | No es programa ni pel pare ni per ell. La crida signal de la línia 25 no s’executa. |
| **34** | ignorar | La programa el pare a línia 30 i el execl no ho canvia |
| **43** | Ignorar | Ho programa el propi procés a la línia 30 |

1. (0,50 punts) Per al procés que executa la línia 43 indica els valors de les variables següents:

|  |  |  |  |
| --- | --- | --- | --- |
| **variable** | **<0** | **=0** | **>0** |
| **pid1** |  |  | **X** |
| **pid2** |  |  | **X** |
| **pid3** |  |  | **X** |

1. (0,50 punts) Quin efecte té en el procés l’execució de la línia 43.

|  |  |  |
| --- | --- | --- |
|  | **Sí/No** | **Justificació** |
| **El procés No sortirà del bucle** | No |  |
| **El procés donarà menys de 2 voltes al bucle** | No |  |
| **El procés donarà exactament 2 voltes al bucle** | Sí | La crida wiat retornarà cada cop que un dels tres fills acabi i el valor de retorn serà el pid del procés fill que s’identifica que ha acabat. Com a molt passarem 3 cops per la condició del bucle una per procés fill. Com que les pipes estan sincronitzades el fill amb pid3 serà l’últim a acabar. |

1. (0,50 punts) Amb relació a les pipes, aquest programa no funciona correctament? Justifica el perquè no funciona i proposa una solució.

Fa falta tancar els canals de les pipes que els processos no utilitzen.

És imprescindible tancar els canals d’escriptura a la pipe ja que aquest fet permetrà rebre un EOF als processos que hi llegeixen i evitar que es puguin produir bloquejos en les lectures de pipes buides.

També s’han de tancar els canals de lectura, ja que aquest fet permetra detectar quan un procés està escrivint informació en una pipe a on cap procés pot llegir, per tant detectar una execució incorrecta, i evitar que es puguin produir bloquejos en les escriptures per buffer ple.

A l’inici de les línies 12, 24, 38 i 43 faria falta posar:

close(fda[0]); close(fda[1]); close(fdb[0]); close(fdb[1]);

1. **Concurrència (3,0 punts)**

En el canal de Suez cada cop més s’ha de controlar el trànsit de vaixells.

En el canal només poden circular vaixells en un sentit. En un moment concret pot haver-hi 3 vaixells, un en cada enclusa del canal. A l’hora d’entrar si hi ha vaixells esperant en el sentit contrari aquests tenen preferència i el vaixell que vol entrar haurà d’esperar.

Al sortir un vaixell, si no queda ningú al canal, es dóna prioritat als vaixells que esperen en sentit contrari al seu.

En els últims temps hi ha hagut problemes i els vaixells s’encallen. Per ajudar-los entren els pràctics del canal. Els pràctics són vaixells petits i poden entrar i sortir del canal quan vulguin i poden ser tants com vulguin.

Mentre hi hagi pràctics en el canal no pot entrar cap vaixell de càrrega. Un cop surt l’últim pràctic (no importa de quin cantó) poden entrar vaixells segons les normes anteriors.

|  |  |
| --- | --- |
| 1 | entrar\_vaixell\_esquerra () |
| 2 | { |
| 3 | if (((ocupat==LL) || ((ocupat==E) && (quants<3) && (esp\_dreta==0))) && |
| 4 | (practics==0)) |
| 5 | { |
| 6 | ocupat=E; |
| 7 | quants++; |
| 8 | } |
| 9 | else |
| 10 | { |
| 11 | esp\_esquerra++: |
| 12 | wait(v\_esquerra); |
| 13 | } |
| 14 | } |
| 15 |  |
| 16 | sortir\_vaixell\_esquerra() |
| 17 | { |
| 18 | quants--; |
| 19 | if (practics==0) |
| 20 | { |
| 21 | if((esp\_esquerra>0) && (esp\_dreta==0)) |
| 22 | { |
| 23 | esp\_esquerra--; ocupat=E; |
| 24 | quants++; |
| 25 | signal(v\_esquerra); |
| 26 | } |
| 27 | else if ((esp\_dreta>0) && (quants==0)) |
| 28 | { |
| 29 | while((quants<3) && (esp\_dreta>0)) |
| 30 | { |
| 31 | esp\_dreta--; |
| 32 | ocupat=D; |
| 33 | quants++; |
| 34 | signal(v\_dreta); |
| 35 | } |
| 36 | }else if (quants==0) |
| 37 | { |
| 38 | ocupat=LL; |
| 39 | } |
| 40 | } |
| 41 | } |

|  |  |
| --- | --- |
| 43 | sortir\_practic() |
| 44 | { |
| 45 | practics--; |
| 46 | if ((practics==0) && (ocupat==E)) |
| 47 | { |
| 48 | if((esp\_esquerra>0) && (esp\_dreta==0)) |
| 49 | { |
| 50 | while((quants<3) && (esp\_esquerra>0)) |
| 51 | { |
| 52 | esp\_esquerra--; |
| 53 | ocupat=E; |
| 54 | quants++; |
| 55 | signal(v\_esquerra); |
| 56 | } |
| 57 | } |
| 58 | else if ((esp\_dreta>0) && (quants==0)) |
| 59 | { |
| 60 | while((quants<3) && (esp\_dreta>0)) |
| 61 | { |
| 62 | esp\_dreta--; |
| 63 | ocupat=D; |
| 64 | quants++; |
| 65 | signal(v\_dreta); |
| 66 | } |
| 67 | }else if (quants==0) |
| 68 | { |
| 69 | ocupat=LL; |
| 70 | } |
| 71 | } |
| 72 | else if ((practics==0) && (ocupat==D)) |
| 73 | { |
| 74 | if((esp\_dreta>0) && (esp\_esquerra==0)) |
| 75 | { |
| 76 | while((quants<3) && (esp\_dreta>0)) |
| 77 | { |
| 78 | esp\_dreta--; |
| 79 | ocupat=D; |
| 80 | quants++; |
| 81 | signal(v\_dreta); |
| 82 | } |
| 83 | } |
| 84 | else if ((esp\_esquerra>0) && (quants==0)) |
| 85 | { |
| 86 | while((quants<3) && (esp\_esquerra>0)) |
| 87 | { |
| 88 | esp\_dreta--; |
| 89 | ocupat=E; |
| 90 | quants++; |
| 91 | signal(v\_esquerra); |
| 92 | } |
| 93 | }else if (quants==0) |
| 94 | { |
| 95 | ocupat=LL; |
| 96 | } |
| 97 | } |
| 98 | } |

1. (0,50 punts) Indica quines variables són variables globals i quines locals.

|  |  |  |
| --- | --- | --- |
| Nom variable | Local/global | JUSTIFICACIÓ |
| practics | global | Controla l’estat del canal i és accedida per més d’un procediment |
| ocupat | global | Ídem. |
| quants | global | Ídem. |
| esp\_esquerra | global | Ídem. |
| esp\_dreta | global | Ídem. |

1. (1,00 punts) Indica per al procediment entrar\_vaixell\_esquerra() en quines línies de codi fa falta posar un wait(mutex) i un signal(mutex). Suposa que es posa al principi de la línia. Si et fa falta reescriu la línia.

|  |  |  |
| --- | --- | --- |
| Línia | wait / signal | Contingut (si et fa falta) |
| 3 | wait(mutex) | wait(mutex); if (.....) |
| 8 | signal(mutex) | signal(mutex); } |
| 12 | signal(mutex) | signal(mutex); wait(v\_esquerra); |

1. (0,50 punts) Indica per a cada semàfor, inclòs el mutex, a quin valor han d’estar inicialitzat i per què s’utilitza.

|  |  |  |
| --- | --- | --- |
| Nom semàfor | valor | JUSTIFICACIÓ |
| mutex | 1 | Exclusió mútua |
| v\_esquerra | 0 | Sincronització |
| v\_dreta | 0 | Sincronització |

1. (0,50 punts) Fes el codi de entrar\_practic ().

entrar\_practic()

{

wait(mutex);

pràctics++;

signal(mutex);

}

Només fa falta incrementar el nombre de pràctics del canal. Com que és una variable global compartida fa falta tocar-la amb exclusió mútua.

1. (0,50 punts) Per quin motiu faria falta utilitzar la instrucció Test&Set en la construcció del codi del semàfor? Justifica la resposta.

El codi del semàfor és una secció crítica ja que accedeix i modifica les variables compartides del comptador del semàfor i la cua per bloquejar als processos.

En una màquina multicore o multi processador , la instrucció test&set permet construir el protocol d’entrada i de sortida d’una exclusió mútua en base a una espera activa que protegeixi la secció crítica del semàfor.

sem\_wait: TS S ;; protocol d’entrada

BNF sem\_wait

return

SECCIÓ CRÍTICA

sem\_signal: mov S, lliure ;; protocol de sortida

1. **Memòria (3 punts)**

Tenim les següents adreces lògiques i físiques d’un procés d’usuari.

Espai lògic del procés Espai físic

(adreça de 13 bits) (adreça de13 bits)

1a) 000 0001100110 010 0001101110

2a) 001 0011001100 001 0001001100

3a) 100 1001110110 011 1000110100

1. (1,50 punts) Mirant la informació anterior, **indica un model de memòria** que podria encaixar amb les dades anteriors (tipus de: **reubicació, particions, contigüitat**). Compte, els espais en blanc de les adreces no tenen per què significar res en concret.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Reubicació Estàtica | | Reubicació Dinàmica | |
|  | Contigu | No Contigu | Contigu | No Contigu |
| Particions Variables |  |  |  | **X** |
| Particions Fixes |  |  |  |  |

Justificació del model de particions

Les particions són variables ja que les adreces base no són totes un mateix múltiple de potència de dos. Suposem que els 3 primers bits de l’adreça lògica indiquen el segment

@ física – desplaçament = @ física bàse

010 0001101110 – 0001100110 = 010 0000001000

001 0001001100 – 0011001100 = 000 1110000000

011 1000110100 – 1001110110 = 010 1110111110

Justificació del model de reubicació

Al ser no contigu només es pot tenir reubicació dinàmica.

Justificació de la contigüitat o no contigüitat

Les adreces lògiques no segueixen el mateix ordre creixent que les físiques. També les adreces base de cada adreça de l’enunciat són diferents

1. (0,50 punts) **Per a l’opció de model de memòria que has contestat** a la pregunta anterior, com quedarà a l’**espai físic** la instrucció següent?

Espai lògic

call @(000 0001100110)

Igual

call @(000 0001100110)

Justificació

Tenim reubicació dinàmica. Les adreces escrites a la memòria física són adreces lògiques del programa.

1. (0,50 punts) **Descriu com és la MMU** d’aquest sistema i si fa **funcions de control i/o de traducció**. Indica els seus registres/taules, els seus camps i els valors que conté per a l’espai del procés anterior.

La MMU és segmentada. Fa les dues funcions: traducció i control. Si es suposa que els segments són de com a molt 1Kbyte

@L = 13 bits

S = 3 bits

D = 10 bits

@F = 13 bits

@base = 13 bits

Límit = 10 bits

Màx seg = 3 bits (8 segments màxim)

Del contingut de la Taula de segments només podem conèixer les adreces base dels segments 0, 1 i 4

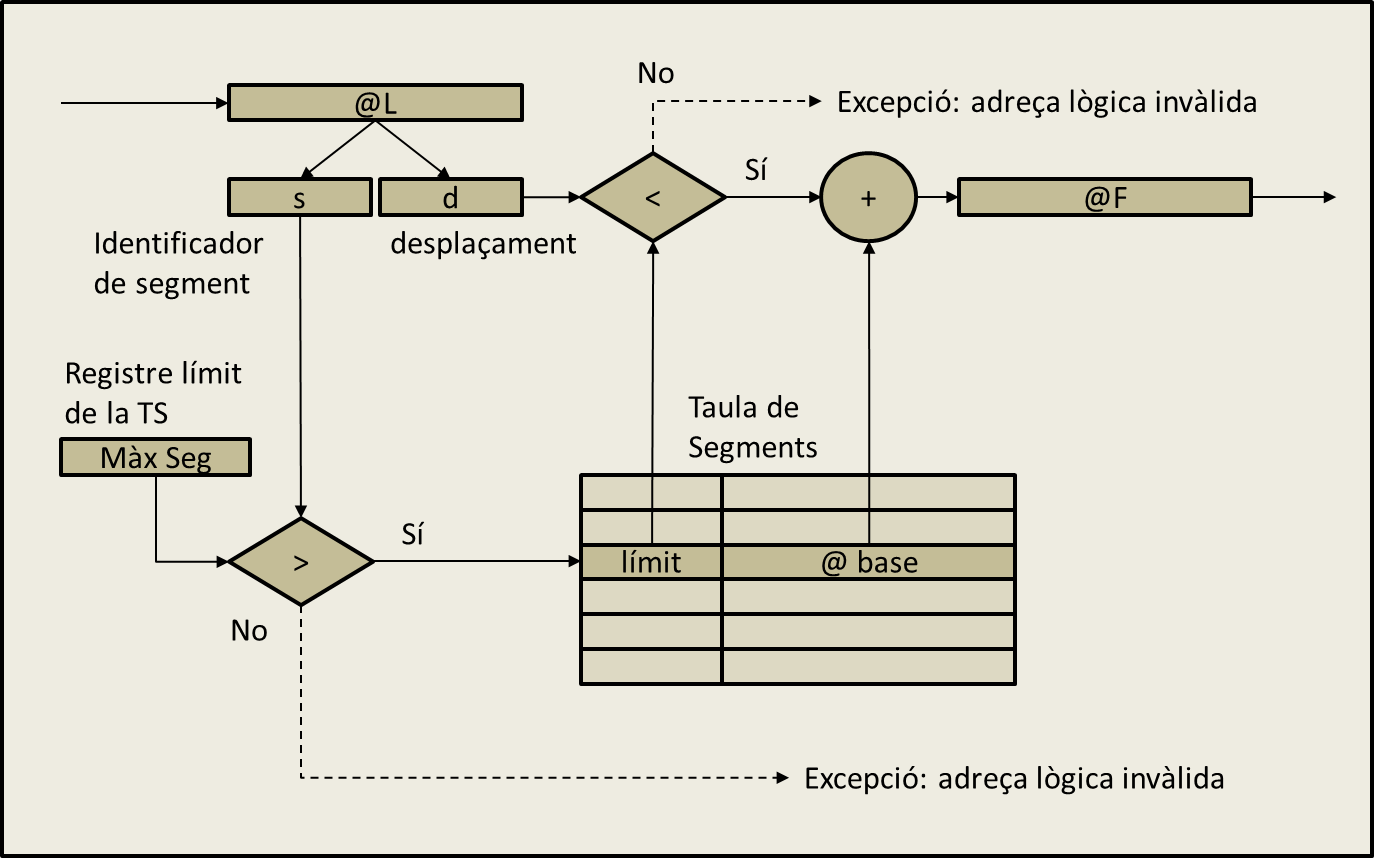
Segment @ física bàse

000 010 0000001000

001 000 1110000000

100 010 1110111110

No tenim informació dels límits ni de si hi ha altres segments



1. (0,50 punts) El model de memòria de l’enunciat pot ser de **Memòria Virtual**?

No, ja que la MMU és segmentada i per tenir MV fa falta tenir paginació.

El fet que el nombre de bytes de memòria física és igual al de la memòria lògica no és determinant.