SISTEMES OPERATIUS

1er Parcial Febrer 2012

2ª Convocatòria

Departament d'Informàtica i Enginyeria Industrial
Escola Universitària Politècnica

Alumne:	**DNI:**

PROBLEMA 1 [3.5 Punts]

En un sistema multiprogramat que disposa d'un processador s'executen 4 processos amb les característiques següents:

Procés	Prioritat	Temps d'arribada	Ràfegues del procés
Procés A	3	0	$4_{CPU}, 2_{E/S}, 3_{CPU}, 2_{E/S}, 1_{CPU}$
Procés B	2	1	$1_{CPU}, 4_{E/S}, 1_{CPU}$
Procés C	1	0	$5_{CPU}, 2_{E/S}, 2_{CPU}, 2_{E/S}, 2_{CPU}$

En cas que 2 ó més processos entrin en la cua de preparats al mateix temps, s'ordenen segons la seva prioritat (prioritat més alta 1, i prioritat més baixa 5).

Es demana:

a) Planificar l'execució dels processos anteriors en un sistema que disposa d'**un processador** i utilitzant l'algorisme primer el mes curt (*SJF*). Utilitzeu la taula següent, indicant l'estat de cada procés (E: en Execució, W: realitzant E/S, P: Preparat, F: Finalitzat, si fan falta més estats afegir-los). [1.25 punts]

Algorisme SJF

	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
A																									
B																									
C																									

b) Planificar l'execució dels processos anteriors en un sistema que disposa d'**un processador** i utilitzant l'algorisme de repartiment de temps (*Round-Robin*) amb un *Quantum* de 3 cicles. Utilitzeu la taula següent, indicant l'estat de cada procés (E: en Execució, W: realitzant E/S, P: Preparat, F: Finalitzat, si fan falta més estats afegir-los). [1.25 punts]

Algorisme Round-Robin

	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
A																									
B																									
C																									

c) Calcular el temps de resposta mig i el temps de retorn mig per l'apartat a. Indiqueu com es podria reduir el temps d'espera mig dels processos. [1.0 punts]

Alumne:

PROBLEMA 2 [3.5 Punts]

a) Escriviu el codi C, amb les crides a sistema necessàries, que utilitzaria l'intèrpret de comandes de Unix per executar aquesta línia de comandes [2.0 punts]:

```
ls | grep prova >> sortida.dat
```

NOTA: Podeu utilitzar les següents crides a sistema:

```
pid_t fork(void);
void exit(int);
int pipe(int desc[2]);
int dup(int fdv);
pid_t wait(int *status)
int close(int fd);
int open(const char *path, int oflag, ... );
```

b) Com detecta el "ls" la finalització anticipada del "grep"? [0.75 punts]

c) Com detecta el "grep" que ha de finalitzar? [0.75 punts]

QÜESTIONS [3 Punts].

Responeu justificadament les qüestions/afirmacions següents. **Escolliu-ne només tres**.

Q1. Indiqueu si el següent programa provoca la utilització dels tres mecanismes pels quals es pot transferir el control d'execució d'un usuari al sistema operatiu [1.0 punts]:

```
int x=10,y=0;
char msg[100];
sprintf(msg,"%d dividit entre %d dóna %d.\n",x,y,x/y);
fd = open("sortida.dat");
write(fd,msg,strlen(msg));
```

Q2. Donat el codi següent expliqueu raonadament quants processos es creen (compteu també el procés pare) i les vegades que la crida a sistema wait s'executa correctament. [1.0 punts]

```
fork();           // 1
if (wait(st) > 0) // 2
    fork();       // 3
```

Q3. Donada la comanda següent:

```
ls –la *.txt | sort > fic.txt
```

Expliqueu raonadament quants processos i quants descriptors de fitxers es necessiten per a la seva execució. [1.0 punts]

Q4. L'execució de comandes en background es poden portar a terme sense la necessitat de crear un nou procés. [1.0 punts]

Q5. Just després de fer un fork per a crear un procés fill, aquest fill hereta el codi i les variables que ha definit el procés pare abans del fork. Per tant els canvis realitzats en aquestes variables pel procés pare, posteriorment a la creació del procés fill, afecten també el procés fill. [1.0 punts]