Alumne:	**DNI:**

PROBLEMA 1 [2.5 Punts]

Donat el programa següent escrit en C i amb crides a sistema de Unix:

```
1  main() {

2      int st, num=0, id;

3      id = fork();
4      if (id!=0) {
5          fork();
6          num+=5;
7      }

8      id = fork();
9      if (id==0) {
10         num+=2;
11         execlp("ls","ls",NULL);
12         exit(1);
13     }
14     else
15         wait(&st);

16     num++;
17     if (id>0)
18         wait(&st);

19     printf("Procés %d acabat amb num=%d.\n",getpid(),num);
20     exit(0);
21 }
```

Assumiu que totes les invocacions a la crida a sistema *fork* acaben correctament.

Es demana que responguеu **justificadament** les preguntes següents:

a) Indiqueu el nombre de processos que es creen durant la seva execució i expliqueu la jerarquia de processos creada. [0.5 punts]

b) Indiqueu les vegades que s'executarà la crida a sistema wait i les vegades que acabarà amb error. [0.5 punts]

c) Indiqueu si algun procés es quedarà en estat de zombie i perquè. [0.5 punts]

d) Indiqueu els missatges que s'imprimiran per pantalla i en quin ordre (assumir que els PIDs dels processos són consecutius, començant en 1000). [0.5 punts]

e) Indiqueu una instrucció on es pugui produir un canvi de modalitat supervisor a modalitat usuari i un canvi de modalitat usuari a modalitat supervisor. Justifiqueu breument la resposta. [0.5 punts]

Alumne:

PROBLEMA 2 [2.5 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	4	0	$3_{CPU}, 2_{E/S}, 1_{CPU}, 2_{E/S}, 2_{CPU}$
Procés B	3	5	$2_{CPU}, 3_{E/S}, 1_{CPU}, 5_{E/S}, 1_{CPU}$
Procés C	1	4	$2_{CPU}, 2_{E/S}, 2_{CPU}$
Procés D	2	2	$3_{CPU}, 5_{E/S}, 1_{CPU}$

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 per prioritats apropiatiu. 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 punt]

Algorisme Prioritats

	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																									
D																									

b) Planificar l'execució dels processos anteriors en un sistema que disposa de **dos processadors** i utilitzant l'algorisme FCFS. 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 punt]

Algorisme FCFS

	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																									
D																									

c) Calcular el temps d'espera mig i el temps de retorn normalitzat mig per l'apartat a. Indiqueu com es podria reduir el temps d'espera mig dels processos. [0.5 punts]

Alumne:

PROBLEMA 3 [2.5 Punts]

Es disposa d'un programa anomenat fer_alarma que conté el codi següent:

```
void rutina_alarm (){
  char buffer[250];

  signal (SIGALRM, rutina_alarm);
  sprintf (buffer, "Ha arribat l'alarma\n");
  write (1, buffer, strlen (buffer));
}

main()
{
  int id;
  int estat;
  char buffer[250];

  signal (SIGALRM, rutina_alarm);
  id = fork();
  switch (id)
  {

      case 0:
              sprintf (buffer, "Procés fill creat\n");
              write (1, buffer, strlen (buffer));
              execlp("app10sec","app10sec",0);
      case -1:
              sprintf (buffer, "Error del fork\n");
              write (1, buffer, strlen (buffer));
              exit(1);
      default:
              wait (&estat);
              sprintf (buffer, "Final de l'aplicació\n");
              write (1, buffer, strlen (buffer));
  }
  exit(0);
}
```

L'executable app10sec correspon al codi:

```
main()
{
  alarm(10);

  sprintf (buffer, "Esperant alarma\n");
  write (1, buffer, strlen (buffer));

  pause();

  sprintf (buffer, "Final del programa\n");
  write (1, buffer, strlen (buffer));

  exit (0);
}
```

Alumne:

Es demana:

a) Expliqueu, pas a pas, l'execució del programa fer_alarma.

b) Suposeu que durant l'execució del programa fer_alarma s'escriuen els missatges següents:

```
Procés fill creat
Error del fork
```

Expliqueu què és el que pot haver passat perquè apareguin aquests missatges i per què.

c) Modifiqueu el codi del programa fer_alarma a fi que l'aplicació app10sec s'executi amb la seva sortida estàndard redireccionada a un fitxer anomenat out.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, ... );
```

Alumne:

QÜESTIONS [2.5 Punts].

Heu d'escollir una opció (la més correcta/completa). Les respostes incorrectes descompten 1/5 de punt.

Test 1: Respecte a la gestió dels senyals als processos, ¿Quina de les següents afirmacions és falsa?:

a) Els programes d'usuari poden definir les rutines de tractament per a totes les senyals que un procés pot rebre.
b) Un senyal es pot enviar a més d'un procés.
c) Es pot enviar una senyal a un fil d'execució.
d) Els processos poden decidir ignorar el tractament d'algunes senyals.

Test 2: Quins recursos **NO** comparteixen els fils d'execució d'un mateix procés:

a) Variables globals/locals, codi, pila i fitxers oberts, entre altres.
b) La pila d'execució, entre altres.
c) Codi i senyals, entre altres.
d) Variables globals/locals i la pila, entre altres.

Test 3: Quins recursos son heretats per un procés fill després de realitzar un fork():

a) Variables i els valors actuals de las variables.
b) Fitxers oberts i rutines tractament de les senyals.
c) Memòria i pila.
d) a i b

Test 4: Quins dels següents afirmacions és certa respecte a la multiprogramació en un sistema amb un sol processador:

a) La multiprogramació sempre permet millorar la velocitat d'execució d'una aplicació seqüencial.
b) La multiprogramació empitjora el rendiment del sistema informàtic.
c) La multiprogramació només permet l'execució d'un procés alhora.
d) Cap de les anteriors.

Test 5: Els sistemes operatius de temps compartit es caracteritzen per:

a) Ser multiprogramats i estar enfocats a treball interactiu
b) Ser multiprogramats i estar enfocats a tasques per lots.
c) Ser monoprogramats i estar enfocat a tasques per lots.
d) Proporcionar un temps de resposta llarg.

Test 6: Quina de les següents instruccions pot provocar els 3 mecanismes de transferència al nucli:

a) a = b/c;
b) write(fd,&buffer,100);
c) getpid();
d) Cap de les anteriors.

Alumne:

Test 7: Quina crida al sistema pot generar un procés zombie si no s'utilitza?

a) Wait.
b) Fork.
c) Exit.
d) Pause.

Test 8: Per quin recurs es pot aplicar el mètode de prevenció basat a prevenir la no apropiació.

a) Gravadores de DVDs.
b) Fitxers oberts en mode escriptura.
c) Impressores.
d) Memòria virtual.

Test 9: Quines de les següents condicions no és necessària perquè es produeixi un interbloqueig:

a) Espera limitada.
b) Exclusió Mútua.
c) Espera circular.
d) No apropiació.

Test 10: Tenim un procés A que després d'executar-se 3 unitats de temps de la seva ràfega de cpu de 4 unitats és tret de la CPU, per executar un procés B que acaba d'entrar en la cua de preparats amb una ràfega de cpu de 3 unitats. El procés A passa a l'estat de preparat. La cua de preparats té més d'un procés. Quin algorisme de planificació s'està utilitzant?

a) SJF.
b) SRT.
c) Per prioritats apropiatiu.
d) Round-Robin amb un quàntum de 3 unitats de temps.