1 Workload 1

1.1 Descripcio del proces

El primer workload prova el rendiment de les diferents politiques de planificacio amb processos intensius en calcul, amb molt poca entrada/sortida. El proces pare crea un fill, que tambe crea un fill (3 procesos en total) i tots executen un gran nombre (igual per a tots) d'iteracions d'un bucle que no fa res (funcio foo()).

Round Robin

Aquesta politica de planificacio reparteix equitativament els recursos entre els procesos, i donat que tots executen el mateix codi tots ocuparan la CPU un nombre molt semblant de ticks, i esperaran a la cua de ready un temps equivalent. El proces idle no entra en cap moment en execucio donat que mai estan tots els procesos bloquejats.

First Come First Served

Amb aquesta politica de planificacio els procesos fills nomes abandonen la CPU quan acaben la seva execucio, i tot i que aixo no afecta al temps total si que fa variar el temps que un proces individual pasa a la cua de ready (el primer fill no esta parat en cap moment, pero el segon ha d'esperar que aquest acabi i el pare ha d'esperar que acabin els 2).

1.2 Codi

```
1
2   void foo(int n){
3
4   }
5
6   void workload1(){
7
8   }
```

2 Workload 2

2.1 Codi

```
1   void workload2(){
2     int pid,pid_f,f,r,i;
3     pid = fork();
4     if(pid == 0){
5       pid_f = fork();
6       if(pid_f == 0){
```

```
 7           for(i = 0; i < 10; ++i)
 8           r = read(0,&buff, 10);
 9           write(1,"1\n",2);
10         }
11         else{
12           for(i = 0; i < 5; ++i)
13           r = read(0,&buff, 20);
14           write(1,"2\n",2);
15         }
16       }
17       else{
18         for(i = 0; i < 20; ++i)
19         r = read(0,&buff, 5);
20         write(1,"p\n",2);
21       }
22     }
```

3 Workload 3