Esercizio 1

Si consideri il seguente codice concorrente:

```
#include <stdio.h>
#include <unistd.h>
int main()
{
   fork();
   fork();
   fork() && fork();
   printf("forked\n");
   return 0;
}
```

Disegnare il relativo albero dei processi, motivandone la costruzione. Si descriva inoltre l'output prodotto dal programma.

Esercizio 2

Quanti processi creano, rispettivamente, i due codici riportati nel seguito?

```
#define N 3
int main()
{
        int i, p;
        for(i=0; i<N; i++) {
           if(p=fork()) break;
        }
}
```

```
#define N 3
int main()
{
        int i, p;
        for(i=0; i<N; i++) {
           p=fork();
        }

}
```

Motivare dettagliatamente la risposta fornita.

Esercizio 3

Si consideri l'insieme dei quattro processi A, B, C, D, che arrivano tutti al tempo 0 e nell'ordine descritto, con tempi di esecuzione in millisecondi:

Processo	Durata CPU–burst
A	20
B	40
C	14
D	6

Calcolare il tempo medio di attesa con scheduling RR e quanto di 10msec. Si consideri ora lo stesso insieme di processi e si supponga che il tempo di context switch sia pari a 2msec e che il processo A esegua prima un CPU–burst da 10msec, quindi un I/O–burst da 40msec e, infine un altro CPU–burst da 10msec, mentre il comportamento degli altri processi resta invariato. Si valuti nuovamente il tempo medio di attesa.

Esercizio 4

Si considerino i processi P_1, P_2 e P_3. Ciascun processo esegue un CPU–burst ed un I/O–burst, quindi nuovamente un CPU–burst ed un I/O–burst ed infine un ultimo CPU–burst. La lunghezza dei burst ed il tempo di arrivo dei processi (in millisecondi) è riportato in tabella:

Processo	CPU–burst_1	I/O–burst_1	CPU–burst_2	I/O–burst_2	CPU–burst_3	Tempo di arrivo
P_1	4	4	3	1	2	0
P_2	1	6	3	3	4	0
P_3	4	2	2	5	2	2

Si disegnino i diagrammi di Gantt che illustrano l'esecuzione dei tre processi utilizzando SJF (non preemptive) ed RR con quanto di tempo pari ad 2 e 4. Se il termine di un servizio di I/O ed un timeout della CPU si verificano nello stesso istante, si assegni la precedenza al processo che ha appena terminato il proprio I/O. Si calcoli il tempo medio di attesa e di turnaround nei tre casi.

Esercizio 5

Si consideri un sistema in cui sono in esecuzione due processi CPU–bound, C_1 e C_2, e quattro processi I/O–bound, O_1, O_2, O_3, O_4. Ciascun processo I/O–bound richiede un'operazione di I/O dopo ogni millisecondo di esecuzione nella CPU. Ciascuna operazione di I/O richiede quattro millisecondi. Si supponga inoltre che vi sia un unico dispositivo di I/O, cosicché richieste multiple in tal senso provochino il formarsi di una coda. Infine, ogni context–switch produrrà un overhead di 0.2 millisecondi. Per arrivare a compimento, ciascun processo CPU–bound richiederà 10 millisecondi, 2 millisecondi saranno invece necessari per i processi I/O–bound. Si faccia l'ipotesi che tutti processi si trovino nella ready queue al tempo 0, nell'ordine C_1, C_2, O_1, O_2, O_3, O_4. Si traccino i diagrammi di Gantt e si calcolino i tempi medi di turnaround utilizzando lo scheduling round–robin e quanti di tempo q_1=10 e q_2=5 millisecondi, rispettivamente.

Esercizio 6

In un sistema di calcolo, la ready queue ad un certo istante è costituita dai processi P_1, con successivo burst di 2 msec, P_2 con burst di 3 msec, P_3 da eseguire per 7 msec e, infine P_4 da 18 msec. Ciascun processo utilizza la CPU per il tempo richiesto e quindi passa ad eseguire un'operazione di I/O della durata di 10 msec; infine, i processi richiedono ulteriori CPU burst ciascuno di ugual lunghezza rispetto al precedente e terminano. Si assuma che le operazioni di I/O vengano eseguite in parallelo. Si disegnino i diagrammi di Gantt che illustrano l'esecuzione dei tre processi utilizzando FCFS, RR con quanto di tempo pari a 2 e SRTF. Si calcoli il tempo medio di attesa e di turnaround nei tre casi.

Esercizio 7

Dati i tre task periodici real–time J_1, con CPU–burst pari a 1msec e periodo di 4msec, J_2 con burst di 2msec e periodo di 6msec, e J_3 con burst di 3msec e periodo di 8msec, si valuti se i tre task sono effettivamente schedulabili utilizzando lo scheduling con priorità proporzionale alla frequenza e lo scheduling EDF e si mostrino i relativi diagrammi di Gantt.

Esercizio 8

Si consideri la seguente situazione, in cui i semafori **s1** e **s2** arbitrano l'accesso concorrente a due risorse, R_1 ed R_2, non condivisibili:

```
P1                          P2
…                           …
wait(s1);                   wait(s1);          valore iniziale s1=1
  <S1>;                       <S2>;            valore iniziale s2=1
signal(s2);                 signal(s1);
```

ed in cui, per errore, è stato scritto **signal(s2)** al posto di **signal(s1)** nel processo **P1**. Come si comportano i processi, indipendentemente dalla loro velocità relativa? Motivare la risposta fornita descrivendone possibili modalità di esecuzione.

Esercizio 9

Disegnare l'albero binario che descrive la valutazione della seguente espressione aritmetica (i nodi interni rappresentano gli operatori unari/binari e le foglie i relativi operandi)

```
(a^2 + (b+c)xd) + (e + fxg)
```

^ operatore di elevamento a potenza

Assumendo che l'accesso ad una variabile o ad una costante sia istantaneo e che per eseguire ciascuna operazione aritmetica si impieghi un'unità di tempo, quanto tempo occorre per valutare l'espressione:

1) sequenzialmente;
2) concorrentemente (in base alla struttura ad albero e supponendo che siano disponibili un numero sufficiente di processori).

Quanto tempo occorre ad eseguire l'espressione, nelle due modalità sopradefinite, supponendo che la somma richieda un'unità di tempo, la moltiplicazione ne richieda due, e l'elevamento a potenza quattro?

Esercizio 10

La seguente coppia di processi condivide la variabile **X**:

```
Processo A          Processo B
     int Y;              int Z;
A1: Y = X*2;        B1:  Z = X+1;
A2: X = Y;          B2:  X = Z;
```

X viene inizializzata al valore 5 prima che inizi l'esecuzione concorrente dei due processi. Quanti sono i possibili valori assunti da **X** al termine dell'esecuzione di entrambi i processi? Successivamente, si supponga che il codice di **A** e **B** venga modificato come descritto nel seguito:

```
Processo A          Processo B
   int Y;              int Z;
   wait(S);            wait(S);
A1: Y = X*2;        B1: Z = X+1;
A2: X = Y;          B2: X = Z;
   signal(S);          signal(S);
```

con S inizializzata ad 1 e **X** a 5 (come nel caso precedente). Quanti sono adesso i possibili valori di **X** quando entrambi i processi hanno terminato la loro esecuzione?

Esercizio 11

Dati i processi **Serial** e **Parallel** come realizzati di seguito, descrivere a parole il risultato della loro esecuzione concorrente.

Dati condivisi

```
semaphore pieno=1, vuoto=0;
int indi=0;
char buff[8];
```

Processo Serial

```
do
{
  wait(pieno);
  buff[indi]=rgb();
  indi++;
  if (indi > 7)
       signal(vuoto);
  else
       signal(pieno);
}while(1);
```

Processo Parallel

```
do
{
  char locb[8];
  wait(vuoto);
  locb=buffer;
  indi=0;
  signal(pieno);
}while(1);
```

Nota: la funzione `rbg()` serve per generare un bit in maniera casuale.

Esercizio 12

Si considerino i processi A, B e C che si sincronizzano attraverso i semafori **Sem1**, **Sem2** e **Sem3** (inizializzati a 0) e che operano sulle variabili condivise **x**, **y** e **z**, inizializzate rispettivamente a 1, 2 e 1.

```
Process A
{
    wait(Sem1);
    z=(x-z)*y;
    x=x+z+y;
    signal(Sem3);
    wait(Sem1);
    x=x+y;
    print(x);
}
```

```
Process B
{
    wait(Sem2);
    x=x+y;
    z=x-z;
    y=(y-z)+x;
    print(y);
    signal(Sem1);
}
```

```
Process C
{
    signal(Sem2);
    wait(Sem3);
    x=x/y;
    y=2*z+x;
    signal(Sem1);
    z=x+z;
    print(z);
}
```

Con quale ordine i processi stampano i valori delle tre variabili? Quali valori vengono stampati? Motivare la risposta data.