Exercici 1
Explica, amb les teves paraules, què és la programació concurrent i quins avantatges i inconvenients presenta.

La programació concurrent es aquella que permet amb només un unic procesador fer diverses tasques simultànies. Una tasca comença para comença un altre para continua l'anterior.

Exercici 2
Quines són les diferències entre programació concurrent, programació paral·lela i programació distribuïda?
La programació concurrent permet executar tasques simultaneas en un mateix procesador com s'explica en la pregunta anterior, en canvi la programació paral·lela permet executar dos processos al mateix temps cada process en un processador. La programació distribuïda en canvi permet executar un programa pero en diversos ordinadors connectats en xarxa .

Exercici 3
Explica què és un servei (daemon). Posa algun exemple.
Es un proces que s'executa en segon pla a l'hora d'iniciar l'ordinador en que l'usuari no hi pot interactuar.

Exercici 4
Explica en quins estats es pot trobar un procés i com pot passar d'un estat a un altre.
Bloquejat,actiu i preparant-se

Exercici 5
Fes un programa en C que generi una estructura de processos amb un pare i 3 fills. Visualitza per cada fill el seu PID i el del pare. Visualitza també el PID del pare de tots.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>
void main(){
	pid_t pid,hijo_pid;
	pid = fork();
	if(pid == -1){
		printf("No se ha podido crear el proceso hijo \n" );
		exit(-1);
	}else if (pid == 0){
		printf("Soy el hijo = 3, Mi padre es: %d , mi pid es: %d\n",getppid(),getpid() );
	}else{
		hijo_pid =wait(NULL);
		pid = fork();
		if(pid == -1){
			printf("No se ha podido crear el proceso hijo \n" );
			exit(-1);
```

```
                }else if (pid == 0){
                        printf("Soy el hijo = 2, Mi padre es %d, mi pid es
%d\n",getppid(),getpid() );
                }else{
                        hijo_pid = wait(NULL);
                        pid =fork();
                        if(pid == -1){
                                printf("No se ha podido crear el hijo 2\n");
                                exit(-1);
                        }else if(pid == 0){
                                printf("Soy el hijo = 1, Mi padre es %d, mi pid es:
%d\n",getppid(),getpid() );
                        }else{
                                hijo_pid = wait(NULL);
                        }
                }
        }
exit(0);
}
```

Exercici 6

Fes un programa en C i crea els pipes necessaris per tal que la comunicació entre un pare i un fill flueixi en tots dos sentits.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>
void main(){
        int fd1[2],fd2[2];
        char buffer[30],buffer2[30];

        pid_t pid;
        pipe(fd1);
        pipe(fd2);
        pid = fork();

        switch(pid){
                case -1://error
                        printf("Ha habido un error\n");
                        exit(-1);
                case 0://Fill
                        close(fd1[1]);
                        read(fd1[0],buffer,20);
```

```
                printf("Fill rep el missatge del pare: %s \n",buffer );
                printf("El fill envia el missatge al pare \n");
                write(fd2[1],"Salutacions del fill", 20);
                break;
        default://Pare
                printf("Pare envia un missatge al fill \n");
                write(fd1[1],"Salutacions del pare",20);
                close(fd2[1]);
                read(fd2[0],buffer,20);
                printf("Pare rep el missatge del fill: %s \n",buffer );
                break;
    }

}
```

Exercici 7

Crea un programa Java fent servir les classes Process i Runtime que en executar-ho des de la línia d'ordres admeti un argument amb l'ordre a executar. Controla la sortida de l'ordre i els errors.

```
import java.io.IOException;
import java.io.*;

public class Ex7RuntimeProcess {
  public static void main(String[] args) throws IOException {
    // TODO Auto-generated method stub

    try {
      String texto;
      Process aEjecutar = Runtime.getRuntime().exec(args[0]);
      BufferedReader reader = new BufferedReader(new
InputStreamReader(aEjecutar.getInputStream()));
      System.out.println("Codi executat:");
      while((texto = reader.readLine()) != null){
        System.out.println(texto);
      }
    } catch (IOException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
    }
  }

}
```

Exercici 8

Crea un programa en Java que llegeixi cadenes des de l'entrada estàndard fins escriure un *. A continuació crea un altre programa que executi l'anterior.

```
import java.io.IOException;
import java.io.*;
public class Ex8 {
   public static void main(String[] args) throws IOException {
      BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
      String texto;
      try {
         do {
            texto = reader.readLine();
         } while (!texto.equals("*"));
      }catch (IOException e){
         e.printStackTrace();
      }
   }
}
```

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.*;

public class Ex8Parte2 {
   public static void main(String[] args) throws IOException {
      // TODO Auto-generated method stub

      try {
         String texto;
         Process aEjecutar = Runtime.getRuntime().exec("java Ex8");
         BufferedReader reader = new BufferedReader(new
InputStreamReader(aEjecutar.getInputStream()));
         System.out.println("Codi executat:");
         while((texto = reader.readLine()) != null){
            System.out.println(texto);
         }
      } catch (IOException e) {
         // TODO Auto-generated catch block
         e.printStackTrace();
      }
   }

}
```