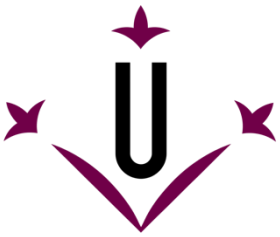

Universitat de Lleida

Escola Politècnica Superior

Grau en Enginyeria Informàtica

Sistemes Operatius

Pràctica 1

Processos, pipes i senyals: EPSlotto

Aleix Segura Paz

Aniol Serrano Ortega

Data entrega: 04/11/2022

Exercici 1

a) Explica de forma breu i aclaridora que fa la funció *fork()*.

La funció *fork()* és una crida al sistema que permet que es dupliqui un procés. La comanda retorna 0 si és el procés fill, major a 0 si és el pare i -1 en cas d'error. El procés fill comparteix el mateix codi, els fitxers oberts del procés pare, el directori en el qual es troba i el valor de les variables en l'instant de la creació.

b) Quin tipus de dades retorna una crida a la funció de creació de processos *fork()*?

pid_t

c) A quin tipus de dades fonamental equival?

Retorna un enter amb signe el que equival a un int.

d) Pot la funció *fork()* retornar el valor -125? I el valor -1?

La funció fork en cas d'error retorna -1, per tant, no pot retornar el valor -125.

e) Quines diferències hi ha amb el grup de funcions *execl()*:

La crida al sistema *fork()* inicia un nou procés còpia del pare, en canvi, l'*execl()* intercanvia el procés actual per un altre usant la mateixa memòria. La comanda *fork()* al ser cridada retrona un valor enter anomenat el pid i la funció no requereix cap paràmetre, en canvi, el grup de comandes execl al no crear cap procés nou no retorna res a més que l'identificador del procés no canvia. Finalment, és destacable el fet de fer ús del recobriment (*execl()*) provoca que el codi màquina, les dades i la memòria utilitzada és reemplaçada pel nou programa.

f) Col·loca els tres blocs de codi següents al forat corresponent del programa que hi ha a continuació (al quadre en blanc que correspongui). Després compila i executa el programa i digues quins missatges s'imprimeixen per la sortida estàndard. Justifica la resposta:

En l'execució del programa primer s'imprimeix l'inici del procés pare amb el seu pid ja, que encara no s'ha fet el *fork()* i no hi ha cap conflicte. Seguidament, el pare es queda en segon pla a l'espera que s'acabi el procés fill en 5 segons i, per tant, s'imprimeixen tots els missatges del procés fill. Un cop el procés fill ha acabat llavors el pare segueix amb l'execució, ja que el fill ha fet un *exit()*. Finalment, el procés pare imprimeix el pid del procés fill amb un codi d'acabament per a reafirmar que efectivament el procés fill ha acabat.

El codi final ha estat el següent:

```c
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/wait.h>
#include <time.h>

#define SEGONSFILL (int)5

char *color_blue = "\033[01;34m";
char *color_red = "\033[01;31m";
char *color_end = "\033[00m";

int main()
{
    int estat,i, exitcode;
    unsigned int llavor;
    char *s;
    pid_t pid,fpid;
    char cadena[100];

    sprintf(cadena, "\n%s[%d] ********** Iniciant PROCES des de %d **********%s\n", color_blue, getpid(), getppid(), color_end);
    if((write(1, cadena, strlen(cadena)))==-1)
        perror("ERROR: write 1");

    pid = fork();

    if(pid==-1) {
        sprintf(cadena, "%s[%d] Error en l'execució del fork.%s\n", color_blue, getpid(), color_end);
        write(2, s, strlen(s));
        exit(0);
    }

    else if(pid==0) {
        sprintf(cadena, "\t%s[%d] ***** Iniciant proces fill de %d ... ***** %s\n", color_red, getpid(), getppid(), color_end);
        write(1, cadena, strlen(cadena));
        llavor = (unsigned int) time(NULL);
        sprintf(cadena, "\t%s[%d] Estableix llavor [%d]%s\n", color_red, getpid(), llavor, color_end);
        write(1, cadena, strlen(cadena));
        srand(llavor); //Establiment de la llavor, diferent en cada execució per assegurar seqüències diferents.

        for(i=0;i<SEGONSFILL;i++) {
            sleep(1);
            sprintf(cadena, "\t%s[%d] El proces fill acabara en %d segons%s\n", color_red, getpid(), SEGONSFILL-i, color_end);
            write(1, cadena, strlen(cadena));
        }

        exitcode = rand()%10; // Genera un nombre aleatori entre 0 i 9.
        sprintf(cadena, "\t%s[%d] El proces fill ha finalitzat retornant amb codi de finalitzacio [%d]!%s\n", color_red, getpid(), exitcode, color_end);
        write(1, cadena, strlen(cadena));
        exit(exitcode);

    }

    else {
        sprintf(cadena, "%s[%d] PROCES PARE suspen execucio en espera de la finalitzacio del fill %d ... %s\n", color_blue, getpid(), pid, color_end);
        write(1, cadena, strlen(cadena));
        if((fpid=wait(&estat)) == -1) {
            sprintf(cadena, "%s[%d] PROCES PARE no te fills ... %s\n", color_blue,
            getpid(), color_end);
            write(1, cadena, strlen(cadena));
            exit(0);
        }

        sprintf(cadena, "%s[%d] PROCES PARE ha rebut acabament proces fill [%d] amb codi [%d] i acaba. %s\n\n", color_blue, getpid(), fpid, WEXITSTATUS(estat),
        color_end);
        write(1, cadena, strlen(cadena));
        exit(0);
    }
}
```

La sortida per consola ha estat la següent:

```
aniol@aniol-VirtualBox:~/Escritorio/Practica 1 - SO$ ./ex1

[4262] ********** Iniciant PROCES des de 3811 **********
[4262] PROCES PARE suspen execucio en espera de la finalitzacio del fill 4263 ...
        [4263] ***** Iniciant proces fill de 4262 ... *****
        [4263] Estableix llavor [1665955695]
        [4263] El proces fill acabara en 5 segons
        [4263] El proces fill acabara en 4 segons
        [4263] El proces fill acabara en 3 segons
        [4263] El proces fill acabara en 2 segons
        [4263] El proces fill acabara en 1 segons
        [4263] El proces fill ha finalitzat retornant amb codi de finalitzacio [6]!
[4262] PROCES PARE ha rebut acabament proces fill [4263] amb codi [6] i acaba.

aniol@aniol-VirtualBox:~/Escritorio/Practica 1 - SO$
```

g) Torna a programar el codi anterior perquè faci el mateix i canviant l'estructura condicional if-else per una estructura switch-case.

Usant l'estructura *switch-case* el codi resultant ha estat el següent:

```c
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/wait.h>
#include <time.h>

#define SEGONSFILL (int)5

char *color_blue = "\033[01;34m";
char *color_red = "\033[01;31m";
char *color_end = "\033[00m";

int main()
{
    int estat,i, exitcode;
    unsigned int llavor;
    char *s;
    pid_t pid,fpid;
    char cadena[100];

    sprintf(cadena, "\n%s[%d] ********** Iniciant PROCES des de %d **********%s\n", color_blue, getpid(), getppid(), color_end);
    if((write(1, cadena, strlen(cadena)))==-1)
        perror("ERROR: write 1");

    pid = fork();

    switch(pid)
    {
        case 1:
            sprintf(cadena, "%s[%d] Error en l'execució del fork.%s\n", color_blue, getpid(), color_end);
            write(2, s, strlen(s));
            exit(0);

        case 0:
            sprintf(cadena, "\t%s[%d] ***** Iniciant proces fill de %d ... ***** %s\n", color_red, getpid(), getppid(), color_end);
            write(1, cadena, strlen(cadena));
            llavor = (unsigned int) time(NULL);
            sprintf(cadena, "\t%s[%d] Estableix llavor [%d]%s\n", color_red, getpid(), llavor, color_end);
            write(1, cadena, strlen(cadena));
            srand(llavor); //Establiment de la llavor, diferent en cada execució per assegurar seqüències diferents.

            for(i=0;i<SEGONSFILL;i++) {
                sleep(1);
                sprintf(cadena, "\t%s[%d] El proces fill acabara en %d segons%s\n", color_red, getpid(), SEGONSFILL-i, color_end);
                write(1, cadena, strlen(cadena));
            }

            exitcode = rand()%10; // Genera un nombre aleatori entre 0 i 9.
            sprintf(cadena, "\t%s[%d] El proces fill ha finalitzat retornant amb codi de finalitzacio [%d]!%s\n", color_red, getpid(), exitcode, color_end);
            write(1, cadena, strlen(cadena));
            exit(exitcode);

        default:
            sprintf(cadena, "%s[%d] PROCES PARE suspen execucio en espera de la finalitzacio del fill %d ... %s\n", color_blue, getpid(), pid, color_end);
            write(1, cadena, strlen(cadena));
            if((fpid=wait(&estat)) == -1) {
                sprintf(cadena, "%s[%d] PROCES PARE no te fills ... %s\n", color_blue,
                getpid(), color_end);
                write(1, cadena, strlen(cadena));
                exit(0);
            }

            sprintf(cadena, "%s[%d] PROCES PARE ha rebut acabament proces fill [%d] amb codi [%d] i acaba. %s\n\n", color_blue, getpid(), fpid, WEXITSTATUS(estat),
            color_end);
            write(1, cadena, strlen(cadena));
            exit(0);
    }
}
```

Per a verificar que la sintaxi és correcta aquest és el *output*:

```
aniol@aniol-VirtualBox:~/Escritorio/Practica 1 - SO$ ./ex1_v2

[4609] ********** Iniciant PROCES des de 3811 **********
[4609] PROCES PARE suspen execucio en espera de la finalitzacio del fill 4610 ...
        [4610] ***** Iniciant proces fill de 4609 ... *****
        [4610] Estableix llavor [1665956715]
        [4610] El proces fill acabara en 5 segons
        [4610] El proces fill acabara en 4 segons
        [4610] El proces fill acabara en 3 segons
        [4610] El proces fill acabara en 2 segons
        [4610] El proces fill acabara en 1 segons
        [4610] El proces fill ha finalitzat retornant amb codi de finalitzacio [5]!
[4609] PROCES PARE ha rebut acabament proces fill [4610] amb codi [5] i acaba.

aniol@aniol-VirtualBox:~/Escritorio/Practica 1 - SO$
```

h) Treball amb nombres aleatoris: en qualsevol de les dues versions (amb if-else o switch-case) feu 3 execucions anotant la llavor establerta (i que es mostra per pantalla) i el codi d'acabament del procés fill. Després comenteu les línies del procés fill que van de llavor = (unsigned int) time(NULL); fins a srand(llavor);. Torneu a fer 3 execucions i anoteu els 3 codis d'acabament. Comenta raonadament els codis d'acabament de les 3 primeres execucions comparats amb els codis d'acabament de les 3 darreres execucions.

Execucions	Amb llavor establerta		Sense llavor establerta
	Llavor establerta	El codi d'acabament del procés fill	El codi d'acabament del procés fill
1	1666041079	6	3
2	1666041127	9	3
3	1666041151	0	3

En el cas en què no treballem amb nombres aleatoris[1] el codi d'acabament sempre és el 3, ja que directament estem retornant el codi d'error que retorna la funció *exit()*. Ara, al no treballar amb nombres aleatoris sempre retorna el mateix nombre.

Finalment, en el cas en què sí que treballem amb nombres aleatoris es genera un nombre aleatori i se li aplica l'operació mòdul [%] 10, el qual restringeix que el número ha d'estar entre 0 i 9.

Exercici 2

Per a la realització d'aquest exercici hem creat 2 arxius, un anomenat loteria.c i l'altre per a fer el recobriment anomenat loteria_2.c. Per a compliar aquests arxius les comandes han de ser les següents respectivament:

gcc -Wall loteria_2.c -o loteria_2

gcc -Wall loteria.c -o loteria

[1] Aleatoris: Suposem nombres aleatoris, però en el context computacional en realitat són nombres semialeatoris els quals tenen distribucions similars a nombres aleatoris reals.

Per a executar aquests arxius, l'instrucció és la següent:

./loteria <numero_enter_aleatori>

A continuació fem una explicació i alguns aclariments del codi:

Loteria.c

Es comença afegint les llibreries típiques necessàries, declarant constants, funcions i variables. Pel que fa a les variables hem creat dos *pipe* bidimensionals per tal de poder tenir 5 fills amb els dos descriptors de fitxers cada un. Tenim un *pipe* per llegir, i l'altre per a escriure, és a dir, un d'anada i un de tornada.

Tot seguit, en el *main*, usem el paràmetre que li passem per execució per a inicialitzar la llavor. Seguidament amb el *fork* creem els N fills i creem els pipes, un per a llegir i un altre per a escriure. Per a acabar fem el recobriment i saltem al programa de loteria_2. Això ho farem per cada un dels fills. Fora del *switch* tenim el gestor de senyals que amb un bucle infinit comprova si es rep un senyal o no.

Finalment, tenim les funcions *ctrlc()* i *ctrl4()*. La primera gestiona l'acabament enviant el senyal SIGTERM a cada un dels fills i imprimeix per terminal el missatge desitjat d'acabament. La segona és el senyal de continuar generant els nombres aleatoris de mida N (5 en aquest cas) i imprimeix per terminal el número premiat.

Loteria_2.c

Ignorem els senyals que tenim, ja que el fill no ho ha de gestionar, ja ho fa el pare.
Fem la generació del nombre aleatori que realitzarà cada un dels fills en la seva posició determinada i escrivim el nombre aleatori generat al *pipe*.