1. ls /proc | grep [0-9]

2. Va afisa toate numerele intre 00 si 99.

3. $(EXE) = reprezinta numele executabilului care va fi

$(OBJS) = reprezinta numele fisierelor obiect pentru compilare

$(LD) = reprezinta numele compilatorului, de ex gcc sau g++

$@ = reprezinta numele tinter curente si se suprapune cu $(EXE)

4. Prin apelul sistemului fork intelegem clonarea procesului respectiv aparand 2

procese, cel principal adica parintele, si cel secundar rezultat din clonare respectiv fiul, care va avea si pid-ul unic la fel ca restul proceselor. Fiul fiind creat isi va continua executia alaturi de procesul parinte in cadrul fork fiecare proces returnand o valoare diferita.

5.

#include <stdio.h>

#include <unistd.h>

#include <signal.h>

#include <stdlib.h>

#include <string.h>

int pfd[2];

int received = 0;

char buffer[200];

pid\_t pidP1, pidP2;

void handler(int *sig*)

{

if(*sig* == SIGUSR1)

received = 1;

}

int main()

{

pipe(pfd);

signal(SIGUSR1, handler);

if(pidP1 = fork())

{

kill(pidP2, SIGUSR1);

}

else

{

if(pidP2 = fork())

{

close(pfd[0]);

char msg[10] = "Hello";

if(received == 1)

write(pfd[1], msg, strlen(msg) + 1);

close(pfd[1]);

}

else

{

close(pfd[1]);

int n;

char readMsg[10];

n = read(pfd[0], readMsg, 10);

printf("%s", readMsg);

close(pfd[1]);

}

}

}

6. I va avea valoarea 1.

Probleme

1.

script(){

for i in "$1"/\*

do

if test -d "$i"

then

if [[ ~$(ls -A $i) ]];

then

echo "$i e gol sau are doar subdirectoare"

fi

script $i

fi

done

}

script $1

2.#include <stdio.h>

#include <unistd.h>

#include <signal.h>

#include <stdlib.h>

#include <string.h>

#define MAX 1024

int pfd[2], received = 0, counter = 0;

char word[64];

pid\_t pid;

void handler(int sig)

{

if(sig == SIGUSR1)

received = 1;

if(sig == SIGALRM)

counter++;

if(counter == 5)

{

printf("Au trecut 5 secunde de la inceperea cautarii!\n");

exit(EXIT\_SUCCESS);

}

}

int main(int argc, char \*argv[])

{

signal(SIGUSR1, handler);

signal(SIGALRM, handler);

pipe(pfd);

if(argc != 3)

{

perror("Eroare la argumente!\n");

exit(EXIT\_FAILURE);

}

pid = fork();

if(pid == 0)

{

close(pfd[1]);

char readBuffer[MAX] = "";

int n;

n = read(pfd[0], readBuffer, sizeof(readBuffer));

strcpy(word, argv[2]);

if(strstr(readBuffer, word))

kill(getppid(), SIGUSR1);

close(pfd[0]);

}

else

{

close(pfd[0]);

alarm(1);

char writeBuffer[MAX] = "", linie[64] = "";

FILE \*f = fopen(argv[1], "r");

while(fscanf(f, "%s", linie) != EOF)

strcat(writeBuffer, linie);

fclose(f);

write(pfd[1], writeBuffer, sizeof(writeBuffer));

close(pfd[1]);

sleep(1);

if(received == 1)

printf("Gasit!\n");

else

printf("Inexistent!\n");

}

}