

**UNIVERSITETI POLITEKNIK I TIRANËS**

**FAKULTETI I TEKNOLOGJISË DHE INFORMACIONIT**

**DEPARTAMENTI I INXHINIERISË INFORMATIKE**

**Punë Laboratori nr. 3**

**Lënda:** Sisteme Operative

**Grupi:** III-B

**Tema:** Krijimi dhe menaxhimi i proceseve dhe thread-eve me ane te thirrjeve te sistemit **fork( )** , **wait( ), pthread\_create( ).**

**Punoi:**  **Pranoi**:

Piro Gjikdhima MSc.Megi Tartari

**Ushtrimi 1**

A screenshot of a computer

Description automatically generatedNderto nje program ne gjuhen C qe perdor thirrjen e sistemit fork( ) per te krijuar dy procese femije nga i njejti proces prind. Prindi P te kete dy procese femije P1 dhe P2.

Ky kod demonstron krijimin e **dy proceseve fëmijë** nga një **proces prind** duke përdorur funksionin **fork()**. Kur **fork()** ekzekutohet për herë të parë, **procesi ndahet** në dy: një prind dhe një fëmijë. Nëse **fork()** kthen 0, kodi ekzekutohet nga **procesi fëmijë**, i cili shtyp **ID-në** e tij dhe të prindit. Në bllokun tjetër, **prindi krijon një proces tjetër fëmijë** me një thirrje të dytë **fork()**. Në fund, **procesi prind** raporton **ID-të** e të dy proceseve fëmijë dhe **ID-në** e tij. **Proceset fëmijë** dhe **prind** ekzekutojnë blloqe të ndryshme të kodit në bazë të vlerës së kthyer nga **fork()**. Në secilin ekzekutim, **output-i mund të jetë i ndryshëm** për shkak të mënyrës se si **skeduleri i sistemit operativ** cakton proceset.

**A screenshot of a computer program

Description automatically generatedOutpute:**

**Ushtrimi 2**

Nderto nje program ne gjuhen C qe perdor thirrjen e sistemit fork( ) per te krijuar nje hierarki prej 3 procesesh te tille qe P2 eshte femija i P1 dhe P1 femija i P.

A screenshot of a computer

Description automatically generated

Ky kod krijon një **hierarki proceseve** duke përdorur thirrje të shumta **fork()**. Fillimisht, **procesi prind** krijon një **proces fëmijë** (**p**). Brenda këtij fëmijë, krijohet një tjetër **proces fëmijë** (**p1**). Më tej, brenda fëmijës së dytë (**p1**), krijohet një **proces tjetër** (**p2**). Çdo **proces fëmijë** shtyp **ID-në** e tij, **ID-në** e prindit dhe, nëse ka, **ID-në** e fëmijës së tij. **Procesi prind** shtyp vetëm **ID-në** e fëmijës së parë që krijoi (**p**). **Hierarkia** ilustron raportet ndërmjet proceseve prind dhe fëmijë.

**Outpute:**

A screenshot of a computer

Description automatically generated

**Ushtrimi 3**

Nderto nje program ne gjuhen C me dy procese prind, femije. Prindi duhet te printoje dy fjali:

1. Prindi ka PID <pid>
2. PID i femijes eshte <PID>.

Procesi femije duhet te printoje dy fjali:

1. Femija ka PID <PID>
2. PID i prindit tim eshte <PID>.

A screenshot of a computer

Description automatically generatedPerdor thirrjen e sistemi wait( ) ne menyre te tille qe te prinohen fillimisht fjalite e procesit child,me pas te prindit.

Ky kod krijon një **proces fëmijë** nga **procesi prind** duke përdorur **fork()**. **Procesi fëmijë** ekzekuton bllokun e kodit brenda **if (p == 0)** dhe shtyp **ID-në** e tij dhe të prindit të tij. Pas kësaj, **procesi prind** përdor **wait(NULL)** për të pritur përfundimin e **procesit fëmijë** dhe më pas shtyp **ID-në** e tij dhe **ID-në** e procesit fëmijë. Kjo siguron që **procesi prind** pritet të përfundojë pas **procesit fëmijë** dhe të dy proceset ekzekutohen në mënyrë të **sinkronizuar**.

**Outpute:**

A screenshot of a computer program

Description automatically generated

**Ushtrimi 4**

Nderto nje program qe krijon nje thread T1 me ane te pthread\_create( ). Procesi kryesor i kalon dy numra thread-it T1. T1 llogarit shumen e ketyre dy numrave dhe ia kthen rezultatin procesit kryesor i cili e printon ne ekran.

A screenshot of a computer

Description automatically generated

Ky kod përdor multithreading për të llogaritur shumën e dy numrave duke përdorur një thread të veçantë.

Në funksionin **main**, përdoruesi jep dy numra, të cilët ruhen në një strukturë të quajtur **ThreadData**. Pastaj krijohet një thread duke përdorur funksionin **pthread\_create**, i cili thërret funksionin **calculate\_sum**. Ky funksion merr strukturën si argument dhe llogarit shumën e dy numrave, duke ruajtur rezultatin në fushën **result** të strukturës. Pasi thread-i përfundon, **pthread\_join** siguron që procesi prind të presë përfundimin e thread-it. Më në fund, rezultati i shumës shtypet në ekran.

**Output:**

A screenshot of a computer

Description automatically generated

**Ushtrimi nga seminari 2**

Ndertoni një program ne gjuhën C qe përdor pipe, ku një proces dërgon një string në një proces të dytë,dhe procesi i dytë konverton stringen ku karakteret kthehen nga shkronja te medha ne te vogla ose anasjelltas me pas dërgohet përsëri tek procesi i parë. Për shembull, nëse procesi i parë dërgon

mesazhin "Hi There", procesi i dytë do të kthejë "hI tHERE". Perdorni dy pipe, një për dërgimin e

mesazhit origjinal nga procesi i parë në procesin e dytë dhe tjetrin për dërgimin e mesazhit të

konvertuar nga procesi i dytë në procesin e parë.

Per te ndertuar programet me lart referojuni programeve shembull me poshte:

**FORK( )**

#include<stdio.h>

#include<unistd.h>

#include<sys/types.h>

int main()

{

pid\_t p;

printf("before fork\n");

p=fork();

if(p==0)

{

printf("I am child having id %d\n",getpid());

printf("My parent's id is %d\n",getppid());

}

else{

printf("My child's id is %d\n",p);

printf("I am parent having id %d\n",getpid());

}

printf("Common\n");

}

**WAIT( )**

#include<unistd.h>

#include<sys/types.h>

#include<stdio.h>

#include<sys/wait.h>

int main()

{

pid\_t p;

printf("before fork\n");

p=fork();

if(p==0)//child

{

printf("I am child having id %d\n",getpid());

printf("My parent's id is %d\n",getppid());

}

else//parent

{

wait(NULL);

printf("My child's id is %d\n",p);

printf("I am parent having id %d\n",getpid());

}

printf("Common\n");

}

**PTHREAD\_CREATE( )**

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<pthread.h>

void \*thread\_function(void \*arg);

int i,j;

int main() {

pthread\_t a\_thread; //thread declaration

pthread\_create(&a\_thread, NULL, thread\_function, NULL);

//thread is created

pthread\_join(a\_thread, NULL); //process waits for thread to finish . //Comment this line to see the difference

printf("Inside Main Program\n");

for(j=20;j<25;j++)

{

printf("%d\n",j);

sleep(1);

}

}

void \*thread\_function(void \*arg) {

// the work to be done by the thread is defined in this function

printf("Inside Thread\n");

for(i=0;i<5;i++)

{

printf("%d\n",i);

sleep(1);

}

}

A black screen with a black background

Description automatically generatedA black screen with a black background

Description automatically generated

Ky kod përdor **dy pipe** dhe një **proces fëmijë** për të konvertuar rastin e shkronjave në një string të dhënë nga përdoruesi.

Në funksionin **main**, krijohen **dy pipe**: një për dërgimin e të dhënave nga prindi tek fëmija dhe një tjetër për kthimin e të dhënave nga fëmija tek prindi. Pasi krijohet një **proces fëmijë** me **fork()**, procesi prind merr një string nga përdoruesi, e dërgon atë te procesi fëmijë përmes **pipe1**, dhe pastaj pret që fëmija të kthejë rezultatin përmes **pipe2**.

Procesi fëmijë lexon të dhënat nga **pipe1**, konverton rastin e shkronjave të string-ut nga **uppercase në lowercase** dhe anasjelltas, dhe e dërgon rezultatin prapa te prindi përmes **pipe2**. Çdo proces mbyll fundin e papërdorur të **pipe-ve** për të parandaluar gabimet e mundshme gjatë komunikimit.

**Output:**

A screenshot of a computer

Description automatically generated

**Ushtrimi nga seminari 2**

Plotesoni programin me poshte ne menyre korrekte ne gjihen C dhe nga ekzekutimi shikoni cfare do te printohet ne ekran ?

int p[2];

char buf1[6]=”Hello”

char buf2[6]=”World”

pipe(p);

if(fork() == 0) {

close(0);

dup(p[0]);

close(p[0]);

close(p[1]);

scanf("%s",buf1);

printf(“%s, buf1);

} else {

close(1);

dup(p[1]);

close(p[0]);

close(p[1]);

printf(“%s, buf2);

}

Kodi I përmirësuar

A black screen with a black background

Description automatically generated

Diferencat midis dy kodeve janë si më poshtë:

1. **Deklarimi i variablave**:
   * Në kodin e parë, variablat **int p[2];** dhe **char buf1[6]="Hello";** janë të deklaruara brenda funksionit main().
   * Në kodin e dytë, variablat **int p[2];** dhe **char buf1[6]="Hello";** janë të deklaruara jashtë funksionit main(), gjë që mund të shkaktojë gabim, pasi variablat nuk janë brenda ndonjë funksioni dhe nuk mund të përdoren ashtu si janë.
2. **Shtypja e printf()**:
   * Në kodin e dytë, ka një gabim në sintaksën e **printf(“%s, buf1);** dhe **printf(“%s, buf2);**, ku do të duhej të ishin **printf("%s", buf1);** dhe **printf("%s", buf2);** për të shtypur variablat në mënyrë të saktë.

A screenshot of a computer

Description automatically generatedKodi krijon një **pipe** që përdoret për të dërguar dhe marrë të dhëna mes procesit prind dhe fëmijës. Procesi prind shkruan "World" në pipe dhe procesi fëmijë e lexon këtë të dhënë dhe e paraqet në ekran si "World". Ky është një shembull i përdorimit të **pipe()**, **fork()**, dhe **dup()** për komunikim ndërmjet proceseve.

**Output:**