**Example/Primjer 1:**

#include <stdio.h>

#include <sys/types.h>

#include <unistd.h>

int main()

{

fork(); //Linija 1

fork(); //Linija 2

fork(); //Linija 3

printf("Pozdrav\n");

return 0;

}

**Example/Primjer 2:**

#include <stdio.h>

#include <string.h>

#include <sys/types.h>

#define MAX\_COUNT 4

#define BUF\_SIZE 10000

int main()

{

pid\_t pid;

int i;

char buf[BUF\_SIZE];

pid\_t p= fork();

pid = getpid();

for (i = 1; i <= MAX\_COUNT; i++) {

sprintf(buf, "Ovo je linija od pid %d, sa vrijednoscu = %d, a ja sam=%d\n", pid, i,p);

write(1, buf, strlen(buf));

}

}

**Example/Primjer 3:**

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#define MAX\_COUNT 20

void main(void)

{

pid\_t pid,rez;

int i;

for (i=1;i<=MAX\_COUNT;i++){

rez=fork();

pid = getpid();

if(rez==-1){

printf ("Rez=%d A- Ovo je proces (PID) %d kojeg nisam uspio kreirati \n",rez, pid);

} else if (rez==0){

printf("Rez=%d B- Ovo je proces Djete/CHILD (PID)=%d, i=%d \n",rez, pid, i);

} else{

printf("Rez=%d C- Ovo je proces TATA/Parent (PID)=%d, i=%d \n",rez, pid, i);

//exit(0);

}

}

}

**Example/Primjer C/C++ 1**

#include <pthread.h>

#include <stdio.h>

#include <stdlib.h>

void\* func(void\* arg)

{

// detach the current thread

// from the calling thread

pthread\_detach(pthread\_self());

printf("Inside the thread\n");

// exit the current thread

pthread\_exit(NULL);

}

**void fun()**

{

pthread\_t ptid;

// Creating a new thread

pthread\_create(&ptid, NULL, &func, NULL);

printf("This line may be printed before thread terminates\n");

// The following line terminates the thread manually

// pthread\_cancel(ptid);

// Compare the two threads created

if(pthread\_equal(ptid, pthread\_self()))

printf("Threads are equal\n");

else

printf("Threads are not equal\n");

// Waiting for the created thread to terminate

pthread\_join(ptid, NULL);

printf("This line will be printed after thread ends\n");

pthread\_exit(NULL);

}

// Main code

int main()

{

fun();

return 0;

}

**Objašnjenje:**

pthread\_detach(pthread\_self());

→ Označava da se ova nit "odvaja" – tj. više je nije moguće “joinati” (čekati s pthread\_join).

Kad završi, njezini resursi se automatski oslobađaju.

printf("Inside the thread\n");

→ Ispisuje poruku da nešto radi unutar dretve.

pthread\_exit(NULL);

→ Završava trenutnu nit.

Glavna funkcija stvara novu nit.

Nova nit se odvoji (detach) i ispiše poruku.

Glavna nit pokušava čekati odvojenu nit (što nije ispravno).

Uspoređuje ID-eve niti.

Na kraju završava proces.

**Example/Primjer C/C++ 2**

**Paralel threads**

#include <pthread.h>

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

**int no\_of\_threads = 4;**

// Funkcija koju svaka nit izvodi

//Thread job to-do

void\* thread\_func(void\* arg)

{

int id = \*((int\*)arg); // dobivamo ID niti (proslijeđen kao argument)

printf("THREAD %d: STARTED!\n", id);

// simulacija nekog posla

// task simulation

for (int i = 0; i < 3; i++) {

printf("Thread %d: i= %d\n", id, i + 1);

sleep(1); // little job simulation...

}

printf("Thread/dretva %d: ending.\n", id);

pthread\_exit(NULL);

}

int main()

{

pthread\_t threads[no\_of\_threads]; //

int thread\_ids[no\_of\_threads]; //

// Stvaranje 4 niti

//Create 4 threads

for (int i = 0; i < no\_of\_threads; i++) {

thread\_ids[i] = i + 1;

int rc = pthread\_create(&threads[i], NULL, thread\_func, &thread\_ids[i]);

if (rc) {

fprintf(stderr, "Greška pri stvaranju dretve/ERROR %d\n", i + 1);

exit(EXIT\_FAILURE);

}

}

// Čekanje da sve niti završe

// Wait until threads are finished

for (int i = 0; i < no\_of\_threads; i++) {

pthread\_join(threads[i], NULL);

}

printf("Sve dretve su završile. END of all threads\n");

return 0;

}