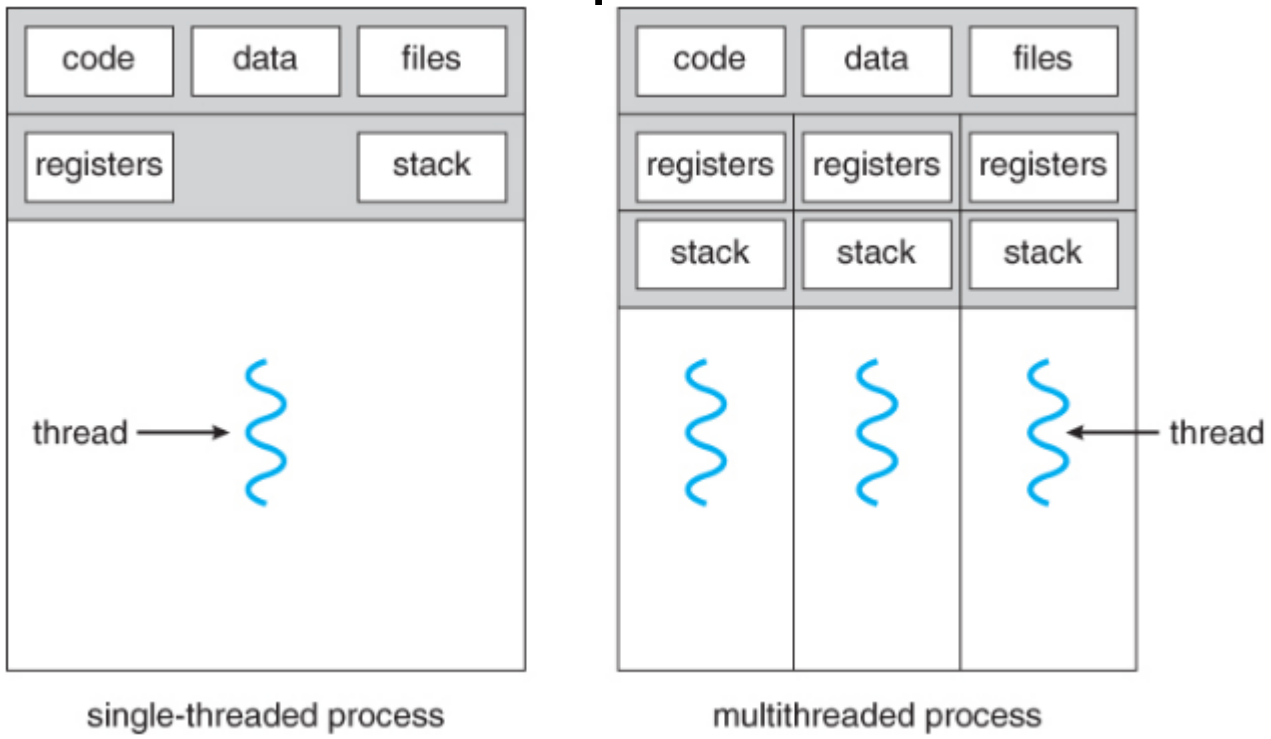
Threads

Threads - תהליכונים

- תהליך אינו יכול לגשת לשום חלק במרחב זיכרון של תהליכים אחרים (אלא אם נגדיר IPC).
- לכל התהליכונים של תהליך מסוים קיימת גישה לאותו מרחב זיכרון.
- ניהול תהליכונים קל יותר ועולה פחות למערכת הפעלה מאשר ניהול תהליכים.

Threads vs Process

- תהליך הנוצר ע"י פעולת FORK – לא חולק שום דבר במשותף עם האבא אלא סה"כ העתק שלו.
- לעומת זאת, תהליכון חולק מרחב זיכרון, קוד, קבצים משותפים וכו'.
- יצירת תכנית עם תהליכים מרובים היא משימה עתירת משאבים ואילו יצירת תהליכונים היא משימה יחסית קלה למערכת הפעלה.



יצירת תכנית עם תהליכונים

- נוסיף לתכנית
```
#include <pthread.h>
```

- קימפול התכנית
```
sysadmin@localhost:~$ gcc thread_program.c -o thread_program -lpthread
```

כאשר :

thread_program – שם התכנית כולל סיומת

-lpthread ייבוא הקובץ pthread.h

- לבסוף – הרצת התכנית ע"י הפקודה
```
sysadmin@localhost:~$ ./thread_program
```

יצירת תהליכון

```
int pthread_create(pthread_t * thread, pthread_attr_t * attr, void * (*start_routine)(void *), void * arg);
```

- הפונקציה מקבלת –

1. thread – מצביע למקום בזיכרון בו יאוחסן מזהה (id) של התהליכון החדש.
2. attr – מאפיינים המתארים את תכונות התהליכון החדש. בד"כ NULL – תהליכון סטנדרטי עם מאפיינים שהם ברירת מחדל.
3. start_routine – מצביע לפונקציה/קוד שאותה התהליכון יצטרך לבצע.
4. args – פרמטרים שיעברו לפונקציה start_routine בעת הפעלתה.

- הפונקציה מחזירה – 0 במקרה של הצלחה. כמו כן מוכנס מזהה id למצביע thread*

  כל ערך אחר שיחזור – מעיד על כישלון ביצירת התהליכון.

יצירת תהליכון

- דוגמא -

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>
void *thread_function(void *args)
{
    sleep(1);
    printf("Printing this from Thread \n");
    return NULL;
}
int main()
{
    pthread_t thread_id;
    printf("Before Thread\n");
    pthread_create(&thread_id, NULL, thread_function, NULL);
    pthread_join(thread_id, NULL);
    printf("After Thread , the pid of thread is: %ld" , thread_id);
    exit(0);
```

```
sysadmin@localhost:~$ gcc thread.c -o thread -lpthread
sysadmin@localhost:~$ ./thread
Before Thread
Printing this from Thread
After Thread , the pid of thread is: 140348859582208sysa
```

סיום תהליכון

```
void pthread_exit(void *retval);
```

- התהליכון שרץ מסיים את פעולתו, ערך סיום שלו יכנס לתוך מצביע retval.
- תהליך ראשי/שאר תהליכונים יוכלו לראות את ערכו מתוך המצביע הנ"ל.

- דוגמאות שימוש -

```
ret1  = 100;
pthread_exit(&ret1);
```

```
pthread_exit(NULL);
```

המתנה לתהליכון מסוים

```
int pthread_join(pthread_t thread, void **retval);
```

- תהליכון ראשי ממתין לסיומו של תהליכון עם מזהה (id) מסוים.
- הפונקציה מקבלת –

1. thread – מזהה תהליכון שממתינים לסיומו.
2. retval – מצביע למקום בזיכרון בו יאוחסן ערך סיום התהליכון שממתינים לו. (אפשר לסמן בNULL אם אין צורך בערך חזרה).

- הפונקציה מחזירה – במקרה של הצלחה – 0.

כמו כן – בהצלחה – מוכנס ערך סיום ב retval.

כל ערך אחר שיחזור מעיד על כישלון.

```
pthread_create(&thread_id, NULL, thread_function, NULL);
pthread_join(thread_id, NULL);
```

pthread_self

```
pthread_t pthread_self(void);
```

- pthread_self() - מחזירה מספר מזהה של התהליכון.(בדומה ל getpid() בתהליכים).
- דוגמא לשימוש -

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>
void *ThreadFun(void *vargp)
{
    sleep(1);
    pthread_t pid_th1 = pthread_self();
    printf("hello , i am thread! my pid is: %ld \n" , pid_th1);
    return NULL;
}
int main()
{
    pthread_t thread_id;
    printf("Before Thread\n");
    pthread_create(&thread_id, NULL, ThreadFun, NULL);
    pthread_join(thread_id, NULL);
    printf("After Thread\n");
    exit(0);
```

```
sysadmin@localhost:~$ nano lab1.c
sysadmin@localhost:~$ gcc lab1.c -o lab1 -lpthread
sysadmin@localhost:~$ ./lab1
Before Thread
hello , i am thread! my pid is: 139760804214528
After Thread
```

pthread_equal

```
int pthread_equal(pthread_t t1, pthread_t t2);
```

- פונקציה מקבלת 2 מזהים של תהליכונים.
- מחזירה ערך כלשהו אם המזהים זהים. אחרת – מחזירה 0.
- דוגמא –

```
#include <pthread.h>
pthread_t tmp_thread;
void* func_one(void* ptr)
{
printf(" tmp_thread id is: %ld\n pthread_self() is %ld",tmp_thread,pthread_self
    if (pthread_equal(tmp_thread, pthread_self()) != 0) {
        printf("equal\n");
    } else {
        printf("not equal\n");
    }
}
int main()
{
    pthread_create(&tmp_thread, NULL, func_one, NULL);
    pthread_join(tmp_thread, NULL);
}
```

```
sysadmin@localhost:~$ gcc lab2.c -o lab2 -lpthread
sysadmin@localhost:~$ ./lab2
 tmp_thread id is: 139729156429568
 , pthread_self() is 139729156429568
equal
```

תהליכונים

- שאלה – מה הפלט של התוכנית?

```
#include <stdio.h>
#include <pthread.h>
#define NTHREADS 20

void *thread(void *vargp)
{
      static int cnt = 0; //
      cnt++;
      printf("%d\n", cnt);
}

int main ()
{

int i;
pthread_t tid;
for(i = 0; i < NTHREADS; i++)
{
      pthread_create(&tid, NULL, thread, NULL);
}
pthread_exit(NULL);
}
```

תהליכונים

- שאלה – מה הפלט של התוכנית?

```
#include <stdio.h>
#include <pthread.h>
pthread_t NewThread;
/*-----------------------------------------------------*/
void *StartHere(void *Message)
{

    if (pthread_equal(NewThread,pthread_self()))
     {
        printf("%s\n",(char *)Message);
    }
    return(NULL);
}
/*-----------------------------------------------------*/
int main(int argc,char *argv[])
{
    pthread_create(&NewThread,NULL,StartHere,"Hello There");


     StartHere("Hello");
     StartHere("There");
     pthread_cond_wait(NewThread,NULL);
 }
```

1. Hello There
2. Hello There
   Hello
   There
3. Hello
   Hello there
   There

תהליכונים

- תרגיל – כתבו תוכנית שיוצרת 2 תהליכונים.

כל תהליכון מדפיס הודעה.

תהליכון 1 ידפיס – thread1

תהליכון 2 ידפיס – thread2

תהליכונים

- פתרון –

מה יקרה ללא ה-join ?

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
void *print_message_function( void *ptr );

main()
{
     pthread_t thread1, thread2;
     char *message1 = "Thread 1";
     char *message2 = "Thread 2";
     pthread_create( &thread1, NULL, print_message_function, (void*) message1);
     pthread_create( &thread2, NULL, print_message_function, (void*) message2);
     pthread_join( thread1, NULL);
     pthread_join( thread2, NULL);
     exit(0);
}

void *print_message_function( void *ptr )
{
     char *message;
     message = (char *) ptr;
     printf("%s \n", message);
}
```

EXEC להתחיל

מטלה - תהליכונים

- תרגיל 1 - כתבו תוכנית שיוצרת תהליכון - מעבירה לפונקציה start_routine פרמטר – מספר שלם כלשהו.

התהליכון מדפיס את המילה "HELLO" ככמות הפעמים של המספר שעבר.

לדוגמא – העברנו לפונקציה מספר שלם 3 – הפלט יהיה

hello hello hello

```
sysadmin@localhost:~$ ./thread
the number is: 3
hello
hello
hello
```