

System calls - process

עבודה עם קומפילר

- פקודה nano – עורך קבצים.
- לדוגמא nano lab2.c
- בתוך הקובץ נוכל לרשום תוכנית בשפת C.
- לדוגמא :

```
#include <stdio.h>
void main(){
printf("hello world"); }
```

```
gcc lab2.c
./a.out
Or
gcc -o lab2 lab2.c
./lab2
```

- הרצה:

תהליכים

- אחת המשימות העיקריות של מע' הפעלה – ניהול תהליכים.
- כל תהליך מריץ תכנית.
- תהליך יכול להיות במצב חסום, מוכן לריצה או במצב ריצה.
- יתכן מצב בו ירוצו כמה תהליכים במקביל.
- מע' הפעלה עוקבת אחרי כל התהליכים שרצים בה ואוכפת את מדיניות הגישה של התהליכים למשאבי מערכת.
- בעת יצירת התהליך הוא יכול – קוד, רשימת משאבים הזמינים לו (זיכרון וכו), הרשאות, מצב מעבד ועוד.
- לכל תהליך יש מספר מזהה משלו (processid (pid
- כל תהליך יכול ליצור תחתיו תהליכי בנים.

סיום תהליך

- סיום רגיל – תהליך מסיים את התכנית ומבצע פינוי משאבים שתפס.
- סיום בעקבות שגיאה – סיום מתוכנן בעקבות שגיאה במהלך ריצת התכנית לדוגמא- תכנית שקיבלה מס פרמטרים לא נכון, דוא"ל שנכשל בשליחה וכו.
- סיום בעקבות שגיאה פטאלית- שגיאה שלא נצפתה מראש ולא היה לה מענה בקוד- התכנית קורסת.
- סיום בעקבות "הריגת" תהליך – מע' הפעלה שולחת בקשה לסיום תהליך כלשהו.

קריאות מערכת לעבודה עם תהליכים

- `pid_t getpid(void)` – פקודה המחזירה pid של התהליך הנוכחי (תהליך בו הפעלנו את הפקודה)

- `pid_t getppid(void)` – פקודה המחזירה pid של תהליך אב לתהליך הנוכחי.

- `pid_t fork(void)` – פקודה זו יוצרת תהליך חדש

- תהליך המפעיל את הפקודה נקרא תהליך אב

- תהליך הנוצר על ידי הפעלת הפקודה נקרא תהליך בן

הערך המוחזר של הפקודה:

- תהליך בן יקבל ערך 0

- תהליך אב יקבל:

- אם תהליך בן נוצר בהצלחה אזי pid של תהליך בן

- אחרת -1

קריאות מערכת לעבודה עם תהליכים

- לעבודה עם קריאות מערכות הקשורות לתהליכים

נוסיף את הספריות הבאות:

```
#include<unistd.h>
```

```
#include <sys/types.h>
```

הפונקציה fork()

- נתונה התכנית הבאה :
מה הפלט ?

```
main() {  
    fork();  
    printf("hello"); }
```

הפונקציה fork()

- תרגיל 1 - נתונה התכנית הבאה :
מה הפלט ?

```
int main()
{
    printf("Hello");
    fork();
    print("bye");
}
```


הפונקציה Fork()

• פתרון תרגיל 1 –

תהליך אבא		תהליך בן
1 printf("hello")		1 x *הבן מתבצע
2 fork()	→	2 x את השורות קוד
3 printf("bye")		3 v לאחר ה- fork

השורה - פלט:

אבא מצבים	{	hello
	}	bye
בן מצבים	{	bye

הפונקציה Fork()

• תרגיל 2 – נתון הקוד הבא – מה הפלט?

```
int main(int argc, char *argv[]) {  
    int forkVal = fork();  
    printf("my pid is: %d\n",  
           my parent pid is: %d\n",  
           fork value: %d\n",  
           getpid(), getppid(), forkVal);  
}
```

הפונקציה Fork()

תרגיל 3 – נתונה התכנית הבאה :
מה הפלט ?

```
int main()
{
    fork();
    fork();
    fork();
    printf("hello.\n");
}
```

הפונקציה Fork()

פתרון תרגיל 3 –

• n - מספר ה forks בתכנית

(יש לשים לב, לפעמים יש תנאים נוספים ואז הנוסחה לא מתאימה)

כמות הילדים שנוצרים : 2^{n-1}

לכן התשובה לתרגיל – כמות ה forks = 3

כמות הילדים שנוצרו $2^2-1=7$

כמות הפעמים שתודפס המילה hello : 8 (7 ילדים + תהליך אב ראשי)

הפונקציה wait

`wait(int *status) pid_t` - משתתף תהליך נכתי עד שתהליך בן יו"ם
את עברו.

* `status` - מצביע למשתנה בו יאוחסן סטטוס נכתי של תהליך ק/י.
ערך 0 - הסן יו"ם

ערך מוכר - `pid` של תהליך שס"ם את עברו
0 -1 - בסיאה

לדומא : (האם מוכר לסיום של אחר מקני)
`wait(null)`

`wait(&status) != -1` // תהליך אב ממתן שבה יסתיים
ומעונין לקבל מידע על הולאקיו

הפונקציה waitpid

$\text{pid-t waitpid}(\text{pid-t pid}, \text{int *status}, \text{int options})$

בקודה שמשהה תהליך עד למצב שתהליך בו עם pid סבסבי יסיים את פעולתו.

משתנה pid יכול לקבל:

- 1. - תחתון לבן כשהו שיסתיים + שפונז - $\text{wait}()$ נקרא
- 0 > תחתון לתהליך עם ה- pid הנתון.

משתנה options יכול לקבל:

WNOHANG - בקודה האם בן מסוים סיים כבר? אם תחתון

אם אין לאי לתחתון.

WUNTRACED - ציחי תהליך שעצב אוב סטאוס מצב לאידוף

הפונקציה waitpid

• הסטטוס שתהליך אב מקבל על תהליך בן מאפשר להפעיל הרבה אפשרויות

#status - יכול להיות על :

- סיום תקין/לא תקין
- סיוקה לסיום
- סטטוס יריאה

בשביל להשתמש בקריאות מערכות הקשורות ל wait נוסף לתוכנית

```
#include <wait.h>
```

דוגמאות שימוש ב wait

לדוגמא:

```
#include <unistd.h>
#include <wait.h> // ספרייה המכילה את פונקציות wait

int main(int argc, char *argv[])
{
    int pid, status;

    if (fork() == 0) // אם תהליך בן - נמצאים pid
        printf("I am the son process and my pid is %d.\n", getpid());
    else
    {
        pid = wait(&status); //
        printf("I am the father process.\n");
        printf("My child with pid %d was exited and returned status %d\n", pid, status);
    }

    return 0;
}
```

תהליך אב - נמצאים שניהם אבא + pid וסטאוס של בן לשניים.

Fork יוצרת

תהליך בן – הבן מתחיל לרוץ ובינתיים תהליך אב מחכה שהבן יסיים ולאחר מכן מדפיס פלט. מהו הפלט !?

דוגמאות שימוש ב wait

פלט לדוגמא –

I am son , my pid is 1111

I am father

My child with pid 1111 was done and return 0

הערה חשובה

wait(&status) - כאשר הבן יסתיים נרצה לדעת עוד פרטים על התהליך.

wait(null) – הבן מסתיים וזהו , שאר המידע לא מעניין את תהליך האב.

דוגמה ליצירת תהליכי בנים

- בדוגמה יהיה תהליך אב יחיד לכל תהליכי הבנים שניצור.
- דוגמא לקוד שמייצר N בנים לאב אחד.

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

int main()
{
    for(int i=0;i<5;i++) // loop will run n times (n=5)
    {
        if(fork() == 0) // יוצר 5 בנים
        {
            printf("[son] pid %d from [parent] pid %d\n",getpid(),getppid());
            exit(0); // בן מוגהים ומסיים
        }
    }
    for(int i=0;i<5;i++) { // loop will run n times (n=5)
        printf("i am pid %d pid %d\n",getpid(),getppid());
        wait(NULL); // מה יקרה אם נוריד את הקטע קוד הנל?
    }
}
```

* מה יקרה אם

נמחק את

`exit(0)`?

יצירת תהליכי בנים בצורת עץ - tree

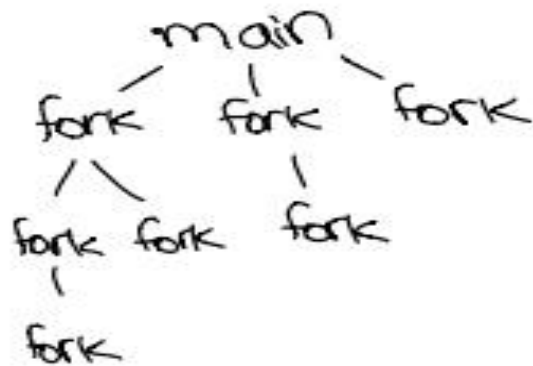
תשובות לשאלות -

- ניצור $2^5 - 1 = 31$ בנים במקום 5 בנים.

- האבא נתפס ב FOR השני וממתין עד שכל חמשת ילדיו ייווצרו. אם נוריד את השורות קוד -> יכול להיות מצב שתהליך האב (התכנית הראשית) תסתיים לפני שכל חמשת הבנים יספיקו להיווצר.

יצירת תהליכי בנים בצורת עץ – tree

```
int main()
{
    fork(); // f1
    fork(); // f2
    fork(); // f3
    printf("hello.\n");
}
```



f1 - יוצר תהליך 1
f2 - יוצרים 2 תהליכים
1 ע"י תהליך אבא (סאט)
1 ע"י התהליך שיוצר ע"י f1

f3 - יוצרים 4 תהליכים
1 ע"י תהליך אבא
1 ע"י תהליך שיוצר א - f1
2 ע"י תהליכים שיוצרו א - f2

סוף 3 fork's $\Leftarrow 7 = 2^3 - 1 = 7$ תהליכי בנים.

מטלה – מעבדה מספר 3

• תרגיל 1 – יש לכתוב תכנית שיוצרת תהליך בן. עבור כל תהליך שרץ יש לבדוק:

• אם התהליך הנוכחי בן -> להדפיס I'm son

• אם התהליך הנוכחי אבא -> להדפיס I am father

מטלה – מעבדה מספר 3

- תרגיל 2 – נתון קטע הקוד הבא : כמה אפשרויות לפלט יש ? יש לפרט מהם הפלטים האפשריים.

```
#include<stdio.h>
#include<sys/wait.h>
#include<unistd.h>

int main()
{
    if (fork()== 0)
        printf("HC: hello from child\n");
    else
    {
        printf("HP: hello from parent\n");
        wait(NULL);
        printf("CT: child has terminated\n");
    }

    printf("Bye\n");
    return 0;
}
```