1. Prerequisites

1.1. Must be able to use basic technology
1.2. Must have mastery of computer concepts.
1.3. Must know data structures
1.4. Must have basic programming skills
1.5. Must be able to develop algorithms
1.6. Must be able to use terminal.
1.7. Must be able to code, compile and run C programs.

2. Prosess Id (PID)

A process runs an instance of the program. If we want to list actively running programs in Unx-based operating systems, we use the "ps" command. If you use "ps x" you will additionally see the "process id" value.

"process id" is called pid. Each process has a unique process id value on the machine it runs on. There are two special process ids in the machines. For example: init(mac lauchd) and scheduler(scheduler). To see these, you must type "ps aux".

Every process has a parent process. The parent process creates child processes. The "getpid()" command is used to get the id value of a process in C language. The "getppid()" command is used to get the id of the parent of a process..

```c
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
    printf("Benim pid = %d.  Anamın-Babamım pid = %d\n", getpid(), getppid());
    return 0;
}
```

```
insect@insect-Lenovo-G50-80: ~/OpSis/prog6
insect@insect-Lenovo-G50-80:~/OpSis/prog6$ gcc main.c -o prog6
main.c: In function 'main':
main.c:5:60: warning: implicit declaration of function 'getpid' [-Wimplicit-function-declaration]
     printf("Benim pid = %d.  Anamın-Babamın pid = %d\n", getpid(), getppid());
                                                            ^
main.c:5:70: warning: implicit declaration of function 'getppid' [-Wimplicit-function-declaration]
     printf("Benim pid = %d.  Anamın-Babamın pid = %d\n", getpid(), getppid());
                                                                    ^
insect@insect-Lenovo-G50-80:~/OpSis/prog6$ ./prog6
Benim pid = 5840.  Anamın-Babamın pid = 5156
insect@insect-Lenovo-G50-80:~/OpSis/prog6$
```

3. Fork Sistem Call

In Unix systems, each process is created with a "fork" system call. Fork: It has meanings such as fork, crossroads.

Fork creates a copy of the process that calls it. This means copying the caller's program control block (code, globals, heap, stact), registers and opened files. The called process returns from the fork with a new pid - this is called the child process - the newly created process copies its parent. The child processes return "0" from the fork call function.

Write and run the prog7 file.

```
insect@insect-Lenovo-G50-80: ~/OpSis/prog7
insect@insect-Lenovo-G50-80:~/OpSis/prog7$ ./prog7
Anayım Ben : pid = 6454
Fork oldu:  Fork id= 6455. Benim pid= 6454.  Anamın pid= 5156
Fork oldu:  Fork id= 0. Benim pid= 6455.  Anamın pid= 6454
insect@insect-Lenovo-G50-80:~/OpSis/prog7$
```

As seen in the figure above, the pid value of the main process is 6454. When the fork operation was performed, the main process first continued its own operation and wrote its values. Then, the child process ran the same codes starting from the fork and wrote its own values. There is no guarantee which processor will receive the fork first.

Write the prog8 d file and run it.

```c
#include <stdio.h>

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

     i = fork();
     if (i == 0) {
       printf("\nCocuk prosess.  getpid() = %d, getppid() = %d\n", getpid(), getppid());
       sleep(5);
       printf("\nUykunun Ardından.  getpid() = %d, getppid() = %d\n",
              getpid(), getppid());
     } else {
         printf("Anayım ben:  Benim pid= %d.  AnamÄ±n pid= %d\n",
          getpid(), getppid());

       printf("\nAnanın işi bitti\n");
     }
    return 0;
}
```

```
insect@insect-Lenovo-G50-80: ~/OpSis/prog8
insect@insect-Lenovo-G50-80:~/OpSis/prog8$ ./prog8
Anayım ben:  Benim pid= 6832.  Anamın pid= 5156

Ananın işi bitti

Cocuk prosess.  getpid() = 6833, getppid() = 6832
insect@insect-Lenovo-G50-80:~/OpSis/prog8$
Uykunun Ardından.  getpid() = 6833, getppid() = 2778
```

As seen above, the parent process has finished its work and the child process has finished its work after a good sleep. Since the mother died before the child's process was completed, the child was left an orphan. In the Linux operating system, the mother of orphaned processes is the "upstart" process (init for Unix). Upstart: It is an application that performs the function of starting tasks and services during startup of the operating system, stopping them during shutdown, and controlling the computer while it is running, and has replaced the traditional init process in Ubuntu as of 6.10.

```
        printf("\nAnanın işi bitti\n");
    }
    return 0;
}
```



```
ct-Lenovo-G50-80: ~/OpSis/prog8
vo-G50-80:~/OpSis/prog8$ ./prog8
 pid= 6832.   Anamın pid= 5156

Ananın işi bitti

Cocuk prosess.  getpid() = 6833, getppid() = 6832
insect@insect-Lenovo-G50-80:~/OpSis/prog8$
Uykunun Ardından.  getpid() = 6833, getppid() = 2778
```

As seen above, the parent process has finished its work and the child process has finished its work after a good sleep. Since the mother died before the child's process was completed, the child was left an orphan. In the Linux operating system, the mother of orphaned processes is the "upstart" process (init for Unix). Upstart: It is an application that performs the function of starting tasks and services during startup of the operating system, stopping them during shutdown, and controlling the computer while it is running, and has replaced the traditional init process in Ubuntu as of 6.10.

Write and run the prog9 file.

```
#include <stdio.h>
int main(int argc, char *argv[])
{
    int i;
    int fv;
        printf("Ana Proses: i= %d  getpid() = %d, getppid() = %d \n",i, getpid(), getppid());
    for (i = 0; i < 3; i++) {
      fv = fork();
      printf("Proses: i= %d  getpid() = %d, getppid() = %d \n",i, getpid(), getppid());
      if (fv == 0) exit(0); // if şartın kapatıp tekrar deneyelim
    }
    return 0;
}
```

```
insect@insect-Lenovo-G50-80: ~/OpSis/prog9
Ana Proses: i= 4195664  getpid() = 6774, getppid() = 6745
Proses: i= 0  getpid() = 6774, getppid() = 6745
Proses: i= 1  getpid() = 6774, getppid() = 6745
Proses: i= 0  getpid() = 6775, getppid() = 6774
Proses: i= 2  getpid() = 6774, getppid() = 6745
Proses: i= 1  getpid() = 6776, getppid() = 6774
Proses: i= 2  getpid() = 6777, getppid() = 2808
insect@insect-Lenovo-G50-80:~/OpSis/prog9$
```

4. Wait Method

The wait command ensures that the main process does not finish until the child processes finish their work. In other words, with the “wait” command, the main process waits for its children to finish their work. The “Wait” command returns the pid value of the child process. The variable that we gave information about how the process ended as a reference to the Wait command is returned numerically. Thus, we can take the pid value of the finished process and the information about why it ended to the main process. To use the Wait Command, you need to call the “sys/wait.h” header.

```c
#include <sys/wait.h>
int main(int argc, char *argv[])
{
    int i, j, status;
    i = fork();
    if (i > 0) {
      j = wait(&status);
      printf("Ana: Cocuk isi Bitirdi.\n");
      printf("  Donen Deger : %d\n", j);
      printf("  Durum:       %d\n", status);
      printf("  WIFSTOPPED:   %d\n", WIFSTOPPED(status));
      printf("  WIFSIGNALED:  %d\n", WIFSIGNALED(status));
      printf("  WIFEXITED:    %d\n", WIFEXITED(status));
      printf("  WEXITSTATUS:  %d\n", WEXITSTATUS(status));
      printf("  WTERMSIG:     %d\n", WTERMSIG(status));
      printf("  WSTOPSIG:     %d\n", WSTOPSIG(status));
    } else {
      printf("Cocuk (%d) Cıkma Istegi Yapti exit(0)\n", getpid());
      exit(0);
    }
    return 0;
}
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/wait.h>
int main(int argc, char *argv[])
{
    int i, j, status;
    i = fork();
    if (i > 0) {
      j = wait(&status);
      printf("Ana: Cocuk Isi Bitirdi.\n");
      printf("  Donen Deger : %d\n", j);
      printf("  Durum:       %d\n", status);
      printf("  WIFSTOPPED:   %d\n", WIFSTOPPED(status));
      printf("  WIFSIGNALED:  %d\n", WIFSIGNALED(status));
      printf("  WIFEXITED:    %d\n", WIFEXITED(status));
      printf("  WEXITSTATUS:  %d\n", WEXITSTATUS(status));
      printf("  WTERMSIG:     %d\n", WTERMSIG(status));
      printf("  WSTOPSIG:     %d\n", WSTOPSIG(status));
    } else {
      printf("Cocuk (%d) Cıkma Istegi Yapti exit(1)\n", getpid());
      exit(1);
    }
    return 0;
}
```

```
insect@insect-Lenovo-G50-80: ~/OpSis/prog12
insect@insect-Lenovo-G50-80:~/OpSis/prog12$ ./prog12
Çocuk (4924) Çıkma Istegi Yapti exit(1)
Ana: Çocuk İşi Bitirdi.
  Donen Deger : 4924
  Durum:       256
  WIFSTOPPED:   0
  WIFSIGNALED:  0
  WIFEXITED:    1
  WEXITSTATUS:  1
  WTERMSIG:     0
  WSTOPSIG:     1
```

```c
#include <stdlib.h>
#include <sys/wait.h>
int main(int argc, char *argv[])
{
    int i, j, status;
    i = fork();
    if (i > 0) {
     printf("Ana Cocugu bekliyor\n");
        sleep(1);
     j = wait(&status);
     printf("Ana: Cocuk Isi Bitirdi.\n");
     printf("  Donen Deger : %d\n", j);
     printf("  Durum:       %d\n", status);
     printf("  WIFSTOPPED:   %d\n", WIFSTOPPED(status));
     printf("  WIFSIGNALED:  %d\n", WIFSIGNALED(status));
     printf("  WIFEXITED:    %d\n", WIFEXITED(status));
     printf("  WEXITSTATUS:  %d\n", WEXITSTATUS(status));
     printf("  WTERMSIG:     %d\n", WTERMSIG(status));
     printf("  WSTOPSIG:     %d\n", WSTOPSIG(status));
    } else {
     printf("Cocuk (%d) Oldurulene Kadar Cikamiyor... Oldur\n", getpid());
     //printf("Child (%d) doing nothing until you kill it\n", getpid());
     while (1) ;
    }
    return 0;
}
```

```
insect@insect-Lenovo-G50-80: ~/OpSis/prog13
insect@insect-Lenovo-G50-80:~/OpSis/prog13$ ./prog13
Ana Çocuğu bekliyor
Çocuk (5307) Oldurulene Kadar Cikamiyor... Oldur
Ana: Çocuk İşi Bitirdi.
  Donen Deger : 5307
  Durum:       15
  WIFSTOPPED:   0
  WIFSIGNALED:  1
  WIFEXITED:    0
  WEXITSTATUS:  0
  WTERMSIG:     15
  WSTOPSIG:     0
insect@insect-Lenovo-G50-80:~/OpSis/prog13$
```

```
insect@insect-Lenovo-G50-80: ~
insect@insect-Lenovo-G50-80:~$ kill 5307
```

5. What Do Those Starting With "W" Mean?

9 / 10

ıl)

process – exit or _exit – returns a non-zero value.

- WEXITSTATUS(stat_val)

The process sent a status code in exit during the destruction process and indicates that value.

- WIFSIGNALED(stat_val)

If the process ends with a signal, it returns a non-zero value. (signal.h)

- WTERMSIG(stat_val)

If the process is finished with a signal, it returns the number of that signal.

- WIFSTOPPED(stat_val)

If the process is stopped, it returns a non-zero value.

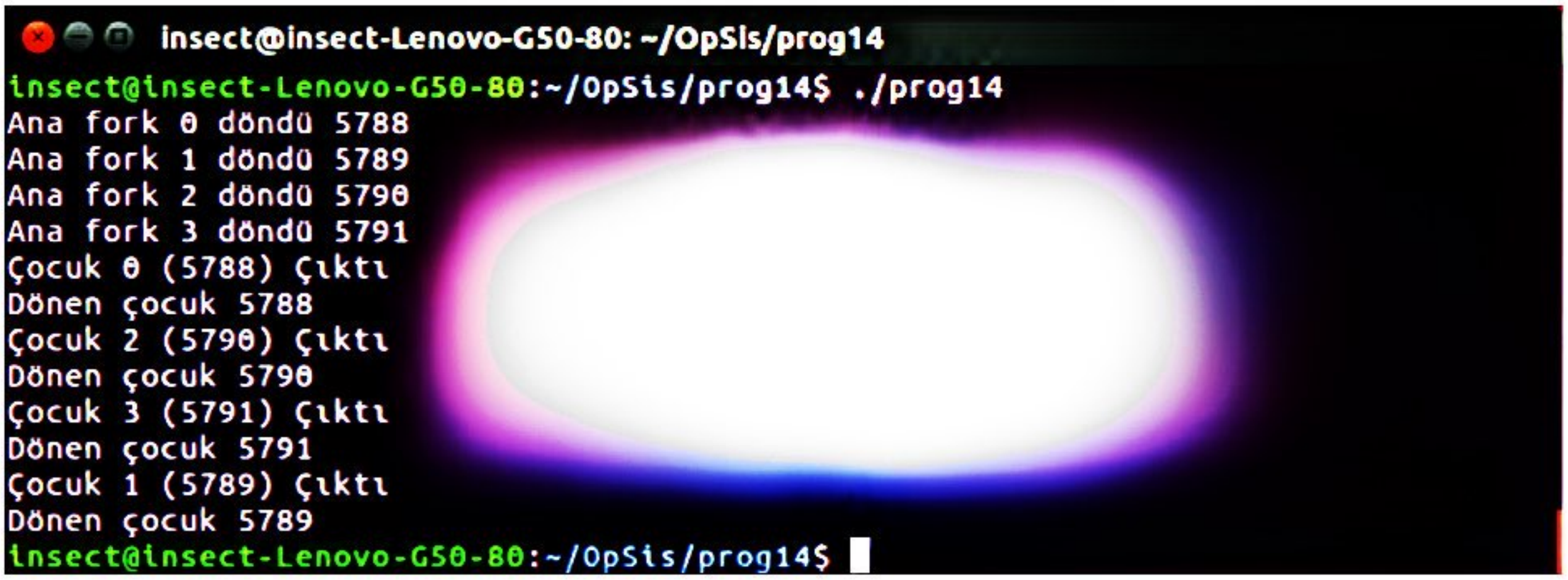
- WSTOPSIG(stat_val)

If the process was stopped by a signal, it returns the number of that signal.

6. Extra Code

Write and run the prog14 code.

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/wait.h>
int main(int argc, char *argv[])
{
    int i, j, status;
    int sleeptime;
    srandom(time(0));
    for (j = 0; j < 4; j++) {
      sleeptime = random()%10;
      i = fork();
      if (i == 0) {
        sleep(sleeptime);
        printf("Cocuk %d (%d) Cikti\n", j, getpid());
        exit(0);
      } else {
        printf("Ana fork %d dondu %d\n", j, i);
      }
    }
    for (j = 0; j < 4; j++) {
      i = wait(&status);
      printf("Donen Cocuk %d\n", i);
    }
    return 0;
}
```



7. Referances

1. **http://web.eecs.utk.edu/~plank/plank/classes/cs360/360/notes/Exec/lecture.html**

2. **http://web.eecs.utk.edu/~plank/plank/classes/cs360/360/notes/Fork/lecture.html**

3. **http://www.comptechdoc.org/os/linux/programming/linux_pgsignals.html**