1. Önkoşullar

1.1. Temel teknoloji kullanabilmeli
1.2. Bilgisayar kavramlarına hakim olmalı.
1.3. Veri yapılarını bilmeli
1.4. Temel programlama yeteneklerine sahip olmalı
1.5. Algoritma geliştirebilmeli

2. Fork ile ne oldu ki?

Fork işlemi çalışıtığı anda bulunduğu programın bir kopyasını oluşturdu ve çalıştığı andan itibaren iki ayrı dal olarak aynı kodu çalıştırdılar. Küçük nüans farklılıkları yanında aynı programı iki defa çağırsan da olabilir diye düşünülebilir. Fakat fork işleminin asıl amacı çalışan ana programın yanında faklı programları ve program parçalarını da çalıştırabiliyor olmasıdır.

3. Exec Komutu

Exec komutu ile başka bir programı programımızın içerisinden çalıştırabiliriz. Exec işlemi sonrasında çalışan prosesin pid değeri ve control bloğu değişmez, proses yaşamını başka bir programı çalıştırmak için devam ettirir.

Prog15 kodunu deneyelim

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

int main(int argc, char *argv[],char ** envp)
{
    char *newargv[3];
      int i;

      newargv[0] = "cat";
      newargv[1] = "main.c"; // program calistigi dosyada herhangi bir dosya
      newargv[2] = NULL;
      //bin dosyasının altına bakalım. Neler var? Neye benziyor?
      //cat programını çalıştırıyoruz
      printf("Ana program: Exec calisti\n");
      i = execve("/bin/cat", newargv, envp);
      perror("exec2: execve failed\n");
      printf("Ana program: Burasi yazacak mi?\n");
    return 0;
}
```

```
insect@insect-Lenovo-G50-80: ~/OpSis/prog15
insect@insect-Lenovo-G50-80:~/OpSis/prog15$ ./prog15
Ana program: Exec calisti
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[],char ** envp)
{
    char *newargv[3];
      int i;

      newargv[0] = "cat";
      newargv[1] = "main.c"; // program calistigi dosyada herhangi bir dosya
      newargv[2] = NULL;
      //bin dosyasının altına bakın. Neler var? Neye benziyor?
      //cat programını çalıştırıyoruz
      printf("Ana program: Exec calisti\n");
      i = execve("/bin/cat", newargv, envp);
      perror("exec2: execve failed\n");
      printf("Ana program: Burasi yazacak mi?\n");
    return 0;
}
insect@insect-Lenovo-G50-80:~/OpSis/prog15$
```

Yukarıda görüldüğü gibi cat programı exec ile çalıştığında ana program terk edildi ve bir daha geri dönülmedi. Geçiş yapılan program bittiğinde program bitti. Bunun anlamı ana program program control bloğuna dallanılan programın değerleri yazıldı. Yer yer buna da ihtiyaç olabilise de genelde istenilmeyen bir durumdur.

Prog16daki iki programıda yazıp derleyelim

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

int main(int argc, char *argv[],char ** envp)
{
    char *newargv[2];
      int i;

      newargv[0] = "Naber";
      newargv[1] = NULL;

      printf("Ana program:getpid: %d  getpppid: %d\n", getpid(), getppid());
      printf("Ana program: Exec calisti\n");
      i = execve("prog16_2", newargv, envp);
      perror("exec2: execve failed\n");
      printf("Ana program: Donus oldu mu?\n");
    return 0;
}
```

```
    i = execve("prog16_2", newargv, envp);
    perror("exec2: execve failed\n");
    printf("Ana program: Donus oldu mu?\n");
  return 0;
}
```

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

int main(int argc, char *argv[])
{
   printf("Alt program:getpid: %d  getpppid: %d\n", getpid(), getppid());
    printf("Ana program mesajÄ±: %s\n", argv[0]);

    return 0;
}
```

```
insect@insect-Lenovo-G50-80: ~/OpSis/prog16
insect@insect-Lenovo-G50-80:~/OpSis/prog16$ ./prog16
Ana program:getpid: 6222  getpppid: 4539
Ana program: Exec calisti
Alt program:getpid: 6222  getpppid: 4539
Ana program mesajı: Naber
insect@insect-Lenovo-G50-80:~/OpSis/prog16$
```

Prog17 deki iki progrmı çalıştırınız.

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

int main(int argc, char *argv[],char ** envp)
{
    char *newargv[2];
      int i;

      newargv[0] = "Naber";
      newargv[1] = NULL;

      printf("Ana program:getpid: %d  getpppid: %d\n", getpid(), getppid());
      int f;
      f= fork();
      if(f==0)
      {
          printf("Ana program: Exec calisti\n");
          i = execve("prog17_2", newargv, envp);
          perror("exec2: execve failed\n");
      }
      else
      {
          printf("Ana program: Donus oldu mu?\n");
      }
    return 0;
}
```

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

int main(int argc, char *argv[])
{
   printf("Alt program:getpid: %d  getpppid: %d\n", getpid(), getppid());
    printf("Ana program mesajÄ±: %s\n", argv[0]);

    return 0;
}
```

```
insect@insect-Lenovo-G50-80: ~/OpSis/prog17
insect@insect-Lenovo-G50-80:~/OpSis/prog17$ ./prog17
Ana program:getpid: 6543  getpppid: 4539
Ana program: Donus oldu mu?
Ana program: Exec calisti
Alt program:getpid: 6544  getpppid: 2823
Ana program mesajı: Naber
insect@insect-Lenovo-G50-80:~/OpSis/prog17$
```

- Exec Komutu eğer herhangi bir nedenden dolayı çalışmazsa geri -1 değerini döndürür. Bu rakam olarak döner ama bunu perror ile yakaladığınızda hata mesajını ekrana yazar.

Prog 18 kodunu yazınız.

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

int main(int argc, char *argv[],char ** envp)
{
    char *newargv[2];
    int i;
    newargv[0] = "Naber";
    newargv[1] = NULL;
    printf("Ana program: Exec calisti\n");
    i = execve("yok", newargv, envp);
    perror("exec: execve hata\n");

    return 0;
}
```

```
insect@insect-Lenovo-G50-80: ~/OpSis/prog18
insect@insect-Lenovo-G50-80:~/OpSis/prog18$ ./prog18
Ana program: Exec calisti
exec: execve hata
: No such file or directory
insect@insect-Lenovo-G50-80:~/OpSis/prog18$
```

Fork işlemi yapıldığında alt programa giden dallanmayı wait komutu ile bekleyebiliriz. Böylece dilersek alt programın işlemini bitirdikten sonra devam edebilir programımız.

Prog20 programlarını inceleyiniz.

```c
#include <stdlib.h>
int main(int argc, char *argv[],char ** envp)
{
    char *newargv[2];
      int i;
      newargv[0] = "Maasallah";
      newargv[1] = "3";
      newargv[2] = NULL;
      printf("Ana program:getpid: %d  getpppid: %d\n", getpid(), getppid());
      int f;
      f= fork();
      if(f==0)
      {
          printf("Ana program: Exec calisti\n");

          i = execve("prog20_2", newargv,envp);
          perror("exec2: execve failed\n");
      }
      else
      {
          printf("Ana: Alt programÄ± bekliyorum...\n");
          wait(&i); // forku bekle
          printf("Ana program: alt program bitirdi\n");
      }
    return 0;
}
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int strtoint( char* charnums)
{

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


    printf("Alt program:getpid: %d  getpppid: %d\n", getpid(), getppid());
    int count= strtoint(argv[1]);
    for (int i = 0; i <= count; ++i) {
          printf("%d %s\n",i, argv[0]);
    }
    return 0;
}
```

```
insect@insect-Lenovo-G50-80: ~/OpSis/prog20
insect@insect-Lenovo-G50-80:~/OpSis/prog20$ ./prog20
Ana program:getpid: 7229  getpppid: 4180
Ana: Alt programı bekliyorum...
Ana program: Exec calisti
Alt program:getpid: 7230  getpppid: 7229
0 Maasallah
1 Maasallah
2 Maasallah
3 Maasallah
Ana program: alt program bitirdi
insect@insect-Lenovo-G50-80:~/OpSis/prog20$
```

Yukarıdaki kodda ana program alt program işini bitirene kadar bekledi ve sonra çıkış yaptı.

4. Exec Faklı Kullanımları

```
int execl(const char *path, const char *arg0, ... /*,(char *) 0 */);

int execv(const char *path, char *const argv[]);

int execle(const char *path, const char *arg0, ... /*, (char *) 0, char *const envp[] */);

int execve(const char *path, char *const argv[], char *const envp[]);

int execlp(const char * file, const char *arg0, ... /*, (char *) 0 */);

int execvp(const char *file, char *const argv[]);
```

Fonksiyonların parametrik benzerlikleri dikkatinizi çekmiştir. exec fonksiyonlarının l'li (execl, execle, execlp) ve v'li (execv, execve, execvp) biçimleri vardır. Bunların l'li biçimleri komut satırı argümanlarını bir liste olarak, v'li biçimleri ise bir dizi olarak alırlar. Ayrıca bazı exec fonksiyon isimlerinin 'e' ya da 'p' harfiyle sonlandırıldığını göreceksiniz. Fonksiyonların e'li biçimleri çevre değişkenlerini (environment variables) de programcıdan istemektedir. p'li biçimler çalıştırılabilen dosyanın yerinin belirlenmesinde PATH çevre değişkenlerini dikkate alırlar.

5. Kaynaklar

1. Wikipedia. *wikipedia.org*. [Online] https://en.wikipedia.org/wiki/Operating_system.

2. http://www.kaanaslan.com/resource/article/display_article.php?id=88

3. tutorialspoint. *www.tutorialspoint.com*. [Online] http://www.tutorialspoint.com/operating_system/os_services.htm.