SİSTEM PROGRAMLAMA
UNIX te İleri Seviye Dosya ve Process Uygulaması

Bölüm-1 : Yeni bir program yazma:

paralel_yazma dosya1.txt dosya2.txt dosya3.txt

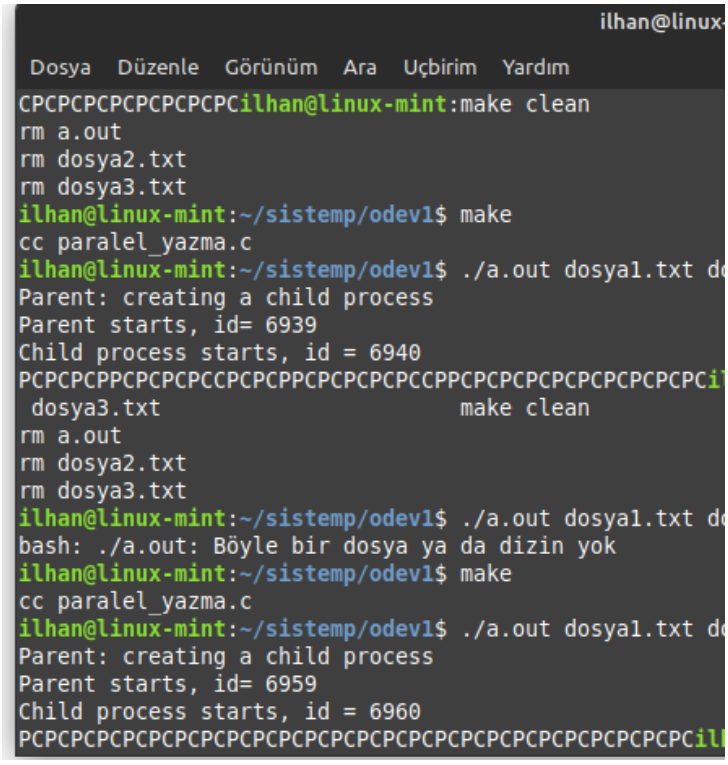
1. Lab3 adında klasör oluşturdum. İçine dosya1.txt adında kaynak dosyayı oluşturdum. pararlel_yazma.c adında kod dosyasını bu klasöre oluşturdum.

2. Programı verilen N değerleri için derleyip birleştiridm.

3.Programı N nin her değeri için birden fazla kez çalıştırdım, dosyalar hiç aynı olmadı.

4. N değerleri için sonuçlar aşağıdadır.

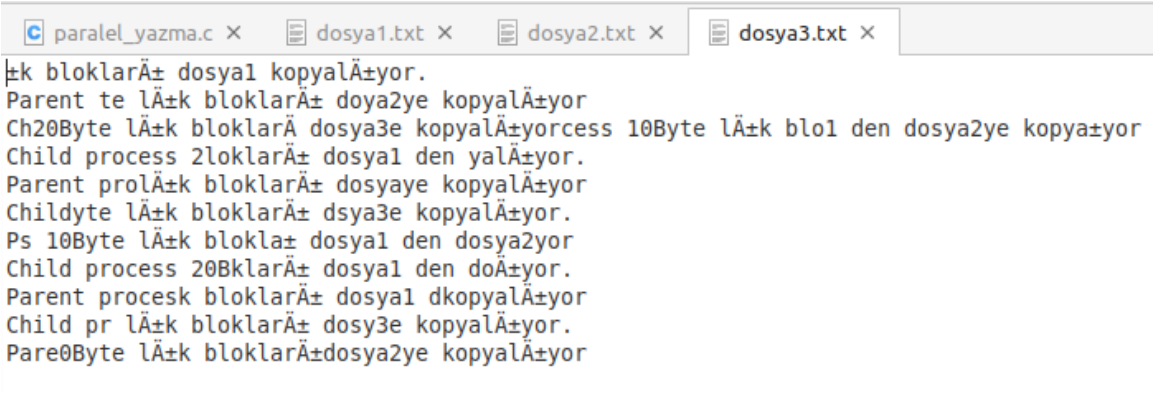
n=100000



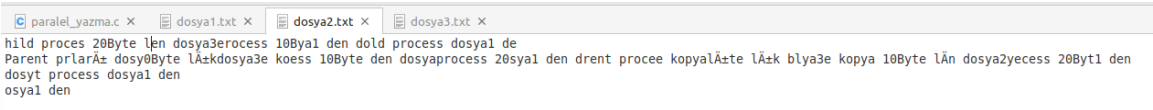

```
pid = fork ();
if (pid == 0)
   {
   printf("Child process starts, id = %d\n",getpid());
   for(;;)
          {

          if (read (fdrd,&ch1, 1) != 1) break;
            read (fdrd,&ch1, 20);

             for (i=0; i<100000;i++);   /* Uzun bir döngü */

write (fdwt3, &ch1, 20);
             write(1,&child,1);
          }
   exit (0);
   }
else
   {
   printf("Parent starts, id= %d\n", getpid());
    for(;;)
          {

          if (read (fdrd, &ch2, 1) != 1) break;

            read (fdrd, &ch2, 10);

             for (i=0; i<100000;i++);   /* Uzun bir döngü */

write (fdwt2, &ch2, 10);
             write(1,&parent,1);
          }
   wait (0);
   }
}
```

n=10000

Dosya3

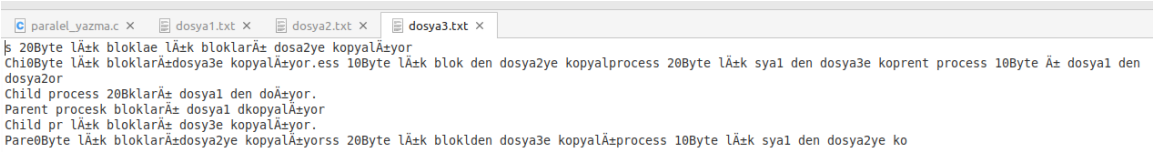


Dosya2

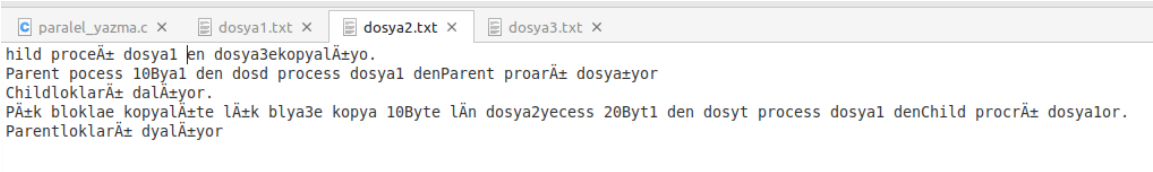


n=5000

Dosya3

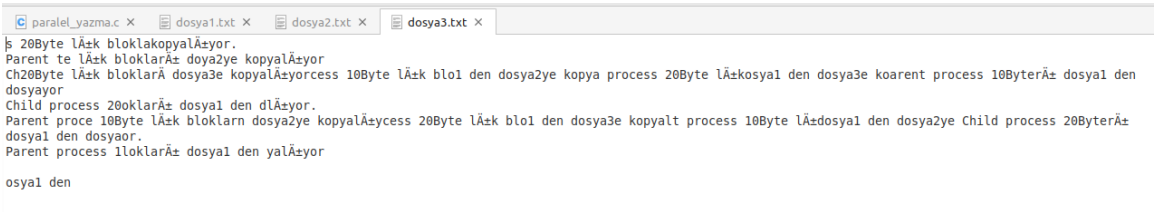


Dosya2

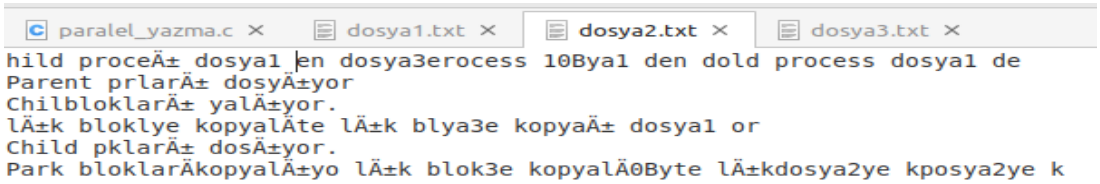


n=10

Dosya3



Dosya2



Dosya2 ve Dosya3 tamamiyle aynı olmadı. Ancak düşük değerlerde dosyalar arasında daha fazla fark oluştu. N 10 değerini alınca Child proses hiç çalışamadı ve dosya3 boş kaldı.

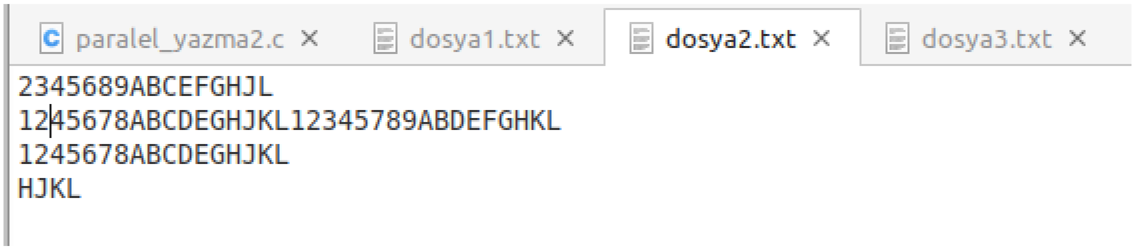
Bölüm-2: Güncelleme

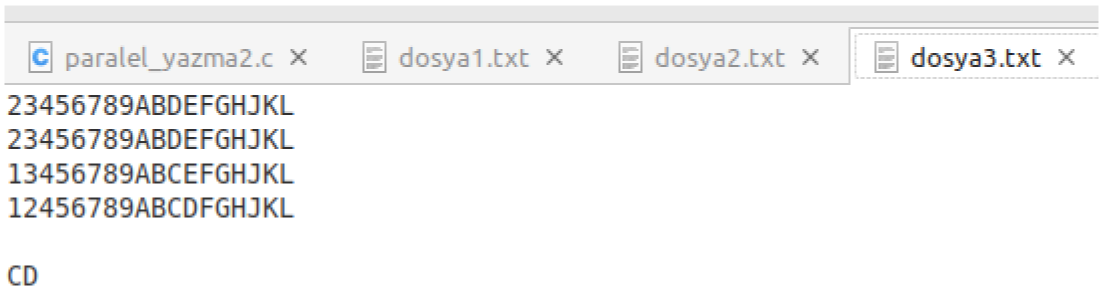
1-2. Dosya farklı isimle kaydedilip kaynak dosya okuması child yaratıldaktan hemen sonra yapıldı

```
if (argc != 4) exit (1);


printf("Parent: creating a child process\n");
pid = fork ();
if ((fdrd = open(argv[1], O_RDONLY)) == -1)exit (1);
if ((fdwt2 = creat(argv[2], 0666)) == -1) exit (1);
if ((fdwt3 = creat(argv[3], 0666)) == -1) exit (1);
if (pid == 0)
   {
   printf("Child process starts, id = %d\n",getpid());
   for(;;)
          {

          if (read (fdrd,&ch1, 1) != 1) break;
            read (fdrd,&ch1, 10);

             //for (i=0; i<10;i++);   /* Uzun bir döngü */

             write (fdwt3, &ch1, 10);
             write(1,&child,1);
          }
   exit (0);
   }
else
   {
   printf("Parent starts, id= %d\n", getpid());
    for(;;)
          {

          if (read (fdrd, &ch2, 1) != 1) break;

            read (fdrd, &ch2, 5);

             //for (i=0; i<10;i++);   /* Uzun bir döngü */

             write (fdwt2, &ch2, 5);
             write(1,&parent,1);
```

3. Beklemeler devre dışı bırakıldı

4. Dosya2 ile Dosya3 içerikleri aynı hemen hemen aynı oldu gibi.





Bölüm-3:

1-2. Yapıldı

3. Sistem işlemi ile child prosesi mesajları karışık olarak ekrana yazdılar. Genellikle sıra doğruydu. Ama yine de bir ratlantısallıktan bahsedilebilir. (Yada yarış)

4. Sistem çağrısından önce for döngüsü ile bir gecikme sağlayınca mesajlar doğru sırada verildi

```
*/
/* burada execv için parametreleri hazırlıyor ..execv exec ailesinin
başka bir üyesi*/
args[0] = "date";
args[1] = "+Date = %D Time = %H:%M";
args[2] = NULL;
for(int i=1;i<50000;i++){}
execv("/bin/date", args);
/*
* eğer exec başarılı ise buraya gelmeyecektir..
*/
perror("exec failed in parent");
```

```
Today's date is:
Date = 12/22/22 Time = 23:06
ilhan@linux-mint:~/sistemp/odev1$ ./a.out
Today's date is:
Date = 12/22/22 Time = 23:06
ilhan@linux-mint:~/sistemp/odev1$ ./a.out
Today's date is:
Date = 12/22/22 Time = 23:06
ilhan@linux-mint:~/sistemp/odev1$ ./a.out
Today's date is:
Date = 12/22/22 Time = 23:06
ilhan@linux-mint:~/sistemp/odev1$ ./a.out
Today's date is:
Date = 12/22/22 Time = 23:06
ilhan@linux-mint:~/sistemp/odev1$ ./a.out
Today's date is:
Date = 12/22/22 Time = 23:06
ilhan@linux-mint:~/sistemp/odev1$ ./a.out
Today's date is:
Date = 12/22/22 Time = 23:09
ilhan@linux-mint:~/sistemp/odev1$ ./a.out
Today's date is:
Date = 12/22/22 Time = 23:09
ilhan@linux-mint:~/sistemp/odev1$
```

5. Sistem çağrısını programın başına yazınca sadece parent işlem ekrana tarihi yazdı. Child hiç çalışmadı. Sistem (parent) prosesi fork ile üretilen bir child prosesten daha öncelikli, hatta childi görmezden geldi anki.

```
char *args[4];

/*
* Bu kodu sadece parent işletecek
*/
/* burada execv için paramet
bir üyesi*/
args[0] = "date";
args[1] = "+Date = %D Time =
args[2] = NULL;
//for(int i=1;i<50000;i++){}
execv("/bin/date", args);
/*
* eğer exec başarılı ise bur
*/
perror("exec failed in paren

exit(1);

if ((pid = fork()) < 0) {
perror("fork failed");
exit(1);
}
if (pid == 0) {
execl("/bin/echo", "echo", "Today's", "date", "is:", 0);
/*
* eğer exec başarılı ise buraya gelmeyecektir.
*/
perror("exec failed in child");

exit(1);
}
```

```
ilhan@linux-mint: ~/sistemp/odev1
Dosya  Düzenle  Görünüm  Ara  Uçbirim  Yardım
Date = 12/22/22 Time = 23:18
ilhan@linux-mint:~/sistemp/odev1$ ./a.out
Date = 12/22/22 Time = 23:18
ilhan@linux-mint:~/sistemp/odev1$ ./a.out
Date = 12/22/22 Time = 23:18
ilhan@linux-mint:~/sistemp/odev1$ ./a.out
Date = 12/22/22 Time = 23:18
ilhan@linux-mint:~/sistemp/odev1$ ./a.out
Date = 12/22/22 Time = 23:18
ilhan@linux-mint:~/sistemp/odev1$ ./a.out
Date = 12/22/22 Time = 23:18
ilhan@linux-mint:~/sistemp/odev1$ ./a.out
Date = 12/22/22 Time = 23:18
ilhan@linux-mint:~/sistemp/odev1$ ./a.out
Date = 12/22/22 Time = 23:18
ilhan@linux-mint:~/sistemp/odev1$ ./a.out
Date = 12/22/22 Time = 23:18
ilhan@linux-mint:~/sistemp/odev1$ ./a.out
Date = 12/22/22 Time = 23:18
ilhan@linux-mint:~/sistemp/odev1$ ./a.out
Date = 12/22/22 Time = 23:18
ilhan@linux-mint:~/sistemp/odev1$ ./a.out
Date = 12/22/22 Time = 23:18
ilhan@linux-mint:~/sistemp/odev1$
```

6. execl: yeni bir işlem imajı oluşturuyor ve parametre olarak liste alıyor
   execv: yeni bir işlem imajı oluşturuyor ancak parametre olarak vektor arumanlar kullanıyor