



**CELAL BAYAR**  
Ü N İ V E R S İ T E S İ

**Hasan Ferdi Turgutlu Teknoloji Fakültesi**

Yazılım Mühendisliği

YZM\_3102 İŞLETİM SİSTEMLERİ

Final Ödevi (PAGING PIPES)

Yrd. Doç. Dr. MANSUR ALP TOÇOĞLU

Mücahit Toktaş 172803036

## Öncelikle Pipe nedir?

Bir process'in outputunu başka bir process'e input olarak yönlendiren File Descriptor'dur.

\*Temel olarak process ler arası iletişim için kullanılır.

\*Bir pipe tek yönlü iletişime izin verir.

\*Veri pipe a yazıldığında aynı sıra ile veriler okunur.

\*Tipik olarak, bir pipe ana proses ile çocuk proses arasında yada tek bir proses içersinde 2 thread arasındaki iletişimi kurmak için kullanılır.

\*Shell içersinde | sembolü bir pipe oluşturur. Örnek olarak ls | less burada bir tane ls için bir tane less için olmak üzere 2 tane çocuk proses üretilmiştir.

\*Shell ayrıca yukardaki örnekte ls nin çıkışını less prosesin girişine bağlar.

\*Bir pipe in veri kapasitesi sınırlıdır.

\*Pipe'a okuma va yazma yapmak için yine read ve write fonksiyonları kullanılır.

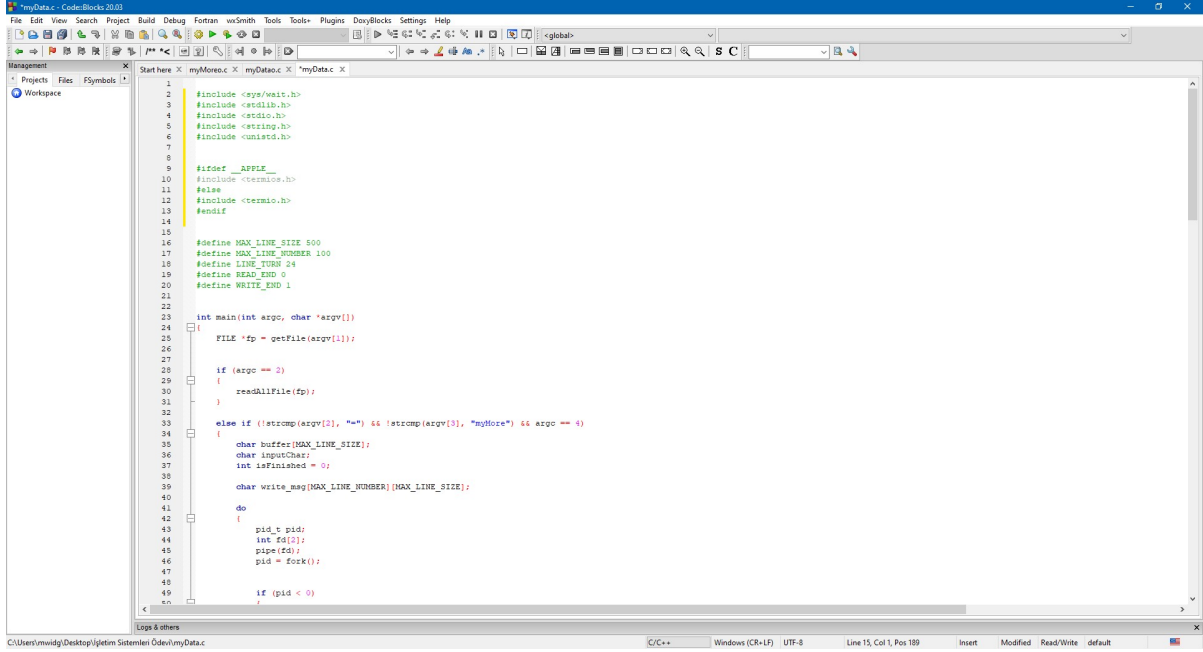
## PROJEMİZİN AMACI VE KAPSAMI

Paging Pipe Projemizin amacı; verilen metin dosyasını myData programı ile okurken, myData programı üzerinden çalışan myMore programı ile aralarında pipe üzerinden veri göndermek ve bunuda 24 satırlık parçalar halinde okumaktır. Daha teknik anlatırsak, kullanıcının girmiş olduğu argüman sayısına göre verilen dokumanı satır satır veya tamamen okumak amaçlanmıştır, bu işlemi gerçekleştirmek içinde C programlama dilinde fork kullanarak programı parent ve child process olarak çalıştırmak gerekmektedir. myData programının çalışması için bir metin dosyasının var olması yeterlidir, myData metin dosyasının tamamını okuyabilmektedir. Ancak programımızın amacı bu değildir, amaç myMore programının myData ile pipe ilişkisi kurması ve bu ilişki ile dosyayı 24 satırda bir kullanıcıdan harf isteyerek okumaya devam etmesi veya okumayı sonlandırmasıdır. Temel gereksinimlerimiz Pipe tanımlama ve fork ile process oluşturmaktır.

Kısaca toparlarsak; myData programımız tek başına çalışabilir ancak myMore programı tek başına çalışamaz. myData programına metin dosyası gönderildiğinde tamamını okur, ama myMore argümanı ile 24 satırlık parçalar halinde okunacaktır. Ayrıca her 24 satırda bir okumaya devam etmek için kullanıcı onayı istenmektedir.

# KODLAR VE KODLARIN AÇIKLAMALARI

## myData.c;



```
1 #include <sys/wait.h>
2 #include <stdlib.h>
3 #include <stdio.h>
4 #include <string.h>
5 #include <unistd.h>
6
7
8
9 #ifdef __APPLE__
10 #include <termios.h>
11 #else
12 #include <termio.h>
13 #endif
14
15
16 #define MAX_LINE_SIZE 500
17 #define MAX_LINE_NUMBER 100
18 #define LINE_TURN 24
19 #define READ_END 0
20 #define WRITE_END 1
21
22
23 int main(int argc, char *argv[])
24 {
25     FILE *fp = getFile(argv[1]);
26
27     if (argc == 2)
28     {
29         readAllFile(fp);
30     }
31     else if (!strcmp(argv[2], "-a") && !strcmp(argv[3], "myMore") && argc == 4)
32     {
33         char buffer[MAX_LINE_SIZE];
34         char inputChar;
35         int isFinished = 0;
36         char write_msg[MAX_LINE_NUMBER][MAX_LINE_SIZE];
37         do
38         {
39             pid_t pid;
40             int fd[2];
41             pipe(fd);
42             pid = fork();
43             if (pid < 0)
44             {
45                 return -1;
46             }
47             if (pid == 0)
48             {
49                 // Child process
50                 // ...
51             }
52             else
53             {
54                 // Parent process
55                 // ...
56             }
57         } while (!isFinished);
58     }
59 }
```

<sys/wait.h> kütüphanesi, wait() fonksiyonunu içermektedir.

Getch() fonksiyonu için MacOS veya Linux Kutuphane seçimi gerçekleştirilir.

"#define MAX\_LINE\_SIZE 500" satır çok uzun olduğunda, sistem tarafından satır ikiye bölünür. Eğer bu limitlenmez ise, satır sayısı kontrolsüz bir şekilde artar.

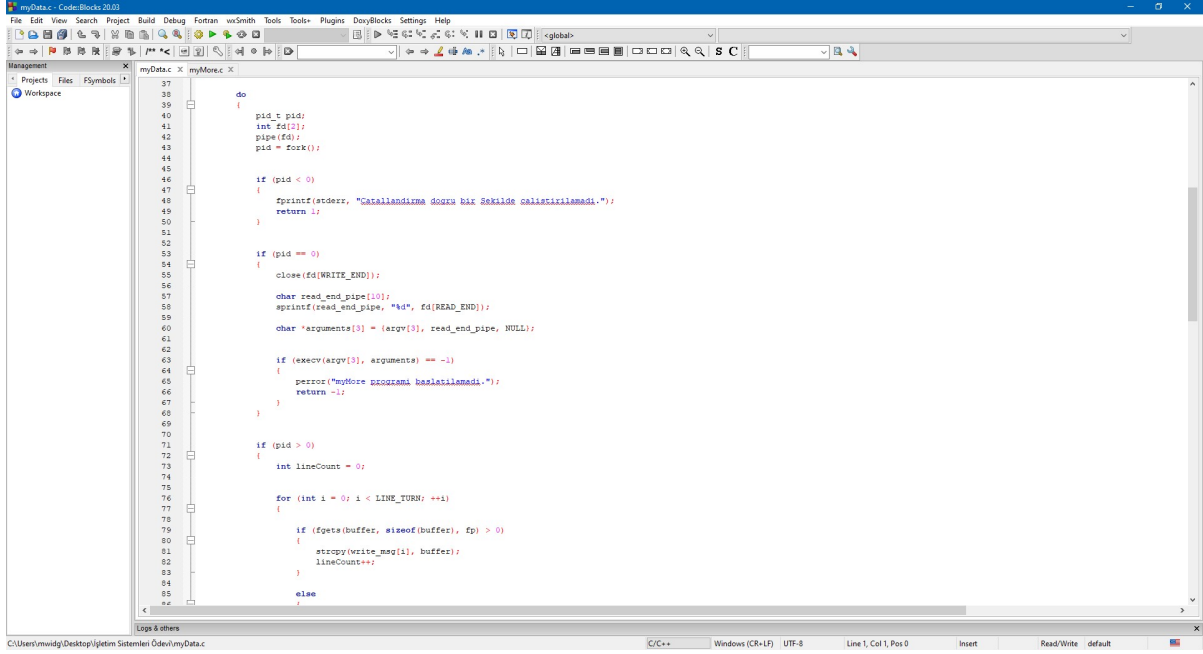
"#define MAXIMUM\_LINE\_NUMBER 100" Dosyadaki maksimum satır sayısını ifade etmektedir.

"#define LINE\_TURN 24" myMore ile basılacak satır sayısını ifade etmektedir.

Int main fonksiyonundaki girdimiz "./myData <filename>" ise bütün dosya okunur, fakat girdimiz

"./myData <filename> = myMore" ise dosyayı 24 satırlık parçalar halinde okur.

"char write\_msg[MAXIMUM\_LINE\_NUMBER][BUFFER\_SIZE];" write\_msg değişkeni pipe aracılığıyla gönderilen değişkeni tutar.



```
37 do
38 {
39     pid_t pid;
40     int fd[2];
41     pipe(fd);
42     pid = fork();
43
44     if (pid < 0)
45     {
46         fprintf(stderr, "Catalandirma dogru bir Sekilde calistirilmedi.");
47         return 1;
48     }
49
50     if (pid == 0)
51     {
52         close(fd[WRITE_END]);
53
54         char read_end_pipe[10];
55         sprintf(read_end_pipe, "%d", fd[READ_END]);
56
57         char *arguments[3] = {argv[3], read_end_pipe, NULL};
58
59         if (execv(argv[3], arguments) == -1)
60         {
61             perror("myMore programi baslatilamadi.");
62             return -1;
63         }
64     }
65
66     if (pid > 0)
67     {
68         int lineCount = 0;
69
70         for (int i = 0; i < LINE_TURN; ++i)
71         {
72             if (fgets(buffer, sizeof(buffer), fp) > 0)
73             {
74                 strcpy(write_msg[i], buffer);
75                 lineCount++;
76             }
77             else
78             {
79                 break;
80             }
81         }
82     }
83 }
84
85 //
```

“pid = fork();” işlemi ile Program çatallandırılır.

“char read\_end\_pipe[10];” read\_end\_pipe değişkeni, pipe'ın okuma ucunu tutar.

“char \*arguments[3] = {argv[3], read\_end\_pipe, NULL};” arguments[3] değişkeni, myMore programına gönderilecek olan parametrelerin dizisini tutar.

“execv” fonksiyonu ile myMore’u çalıştırmaktayız.

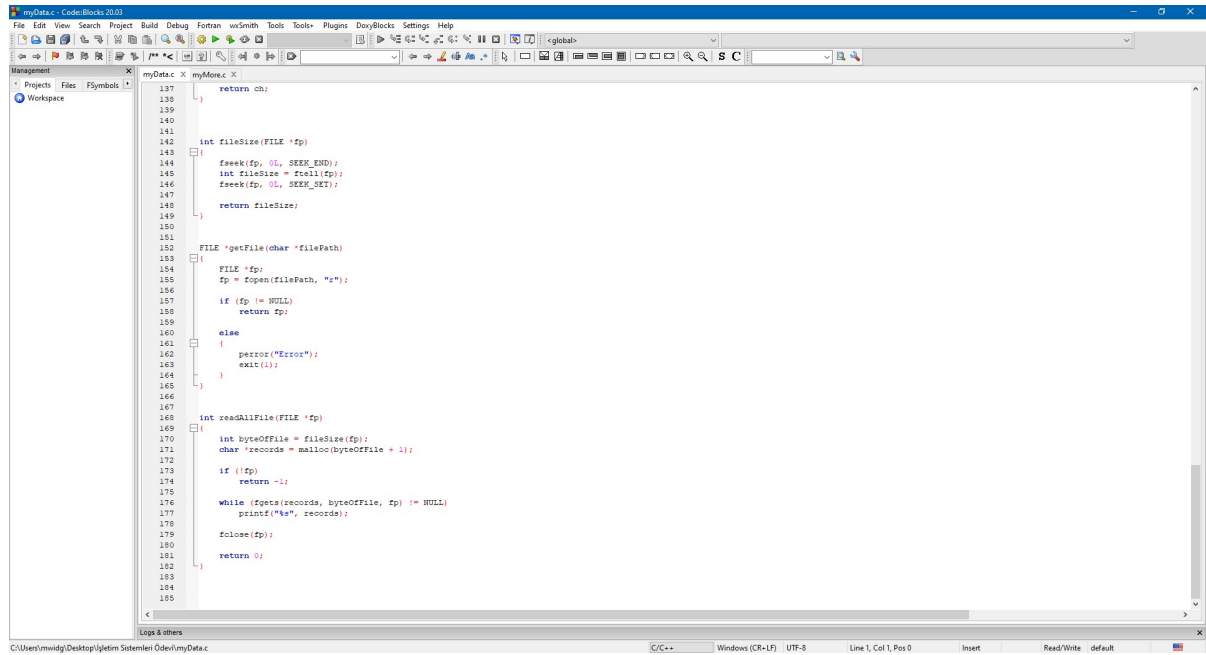
“for (int i = 0; i < LINES\_PER\_TURN; ++i)” bu for döngüsü 24 satır okur, eğer 24 satırdan daha az kaldıysa tamamını okur

“if (fgets(buffer, sizeof(buffer), fp) > 0)” Satırları okuyan if koşulumuz.



“int Getch(void)” fonksiyonumuz, kullanıcının devam etmek için girdiği tuşu kontrol eder, Linux’ta termio.h kütüphanesi ve MacOS’ta da termios.h kütüphanesi bu fonksiyon için kullanılır.

“int fileSize(FILE \*fp)” fonksiyonunun amacı, dosya boyutunu vermektir. (byte bazında)

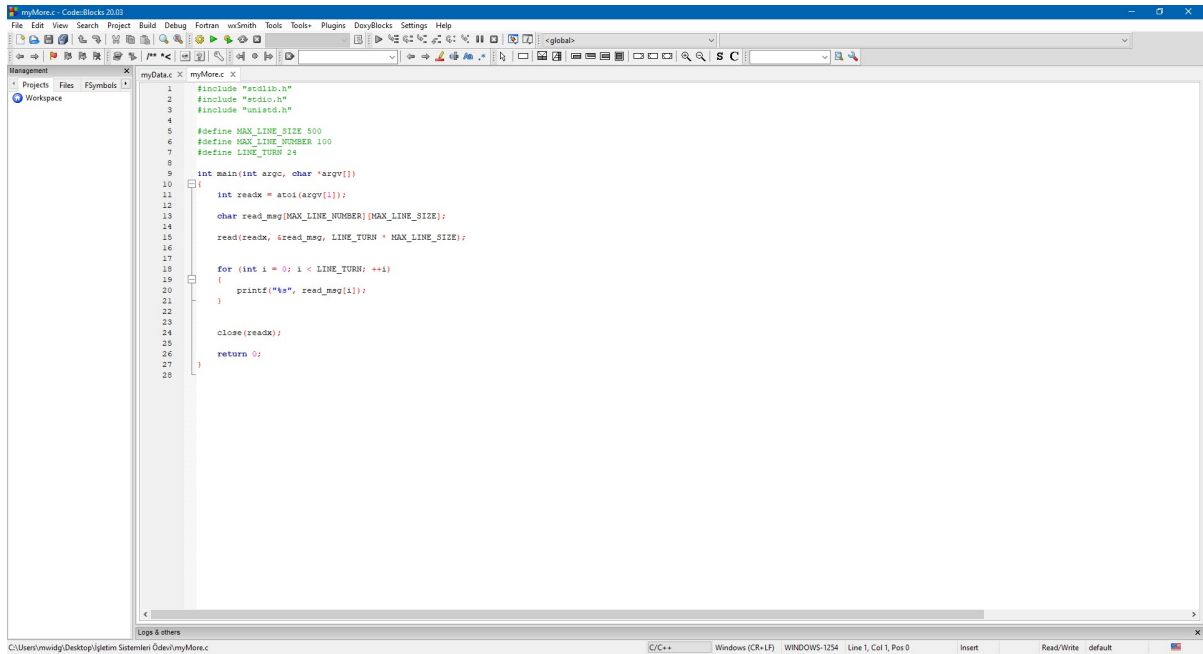


```
137 |     return ch;
138 | }
139 |
140 |
141 |
142 | int fileSize(FILE *fp)
143 | {
144 |     fseek(fp, 0L, SEEK_END);
145 |     int fileSize = ftell(fp);
146 |     fseek(fp, 0L, SEEK_SET);
147 |
148 |     return fileSize;
149 | }
150 |
151 |
152 | FILE *getFile(char *filePath)
153 | {
154 |     FILE *fp;
155 |     fp = fopen(filePath, "r");
156 |
157 |     if (fp == NULL)
158 |         return fp;
159 |
160 |     else
161 |     {
162 |         perror("Error");
163 |         exit(1);
164 |     }
165 | }
166 |
167 |
168 | int readAllFile(FILE *fp)
169 | {
170 |     int byteOfFile = fileSize(fp);
171 |     char *records = malloc(byteOfFile + 1);
172 |
173 |     if (!fp)
174 |         return -1;
175 |
176 |     while (fgetc(records, byteOfFile, fp) != NULL)
177 |         printf("%s", records);
178 |
179 |     fclose(fp);
180 |
181 |     return 0;
182 | }
183 |
184 |
185 |
```

“FILE \*getFile(char \*filePath)” fonksiyonu, dosyanın işaretçisini vermektedir.

“int readAllFile(FILE \*fp)” Fonksiyonunun amacı da dosyayı okumak ve ekrana basmaktır.

## myMore.c;



```
1 #include "stdlib.h"
2 #include "stdio.h"
3 #include "unistd.h"
4
5 #define MAX_LINE_SIZE 100
6 #define MAX_LINE_NUMBER 100
7 #define LINE_TURN 24
8
9 int main(int argc, char *argv[])
10 {
11     int readx = atoi(argv[1]);
12
13     char read_mag[MAX_LINE_NUMBER][MAX_LINE_SIZE];
14     read(readx, &read_mag, LINE_TURN * MAX_LINE_SIZE);
15
16     for (int i = 0; i < LINE_TURN; ++i)
17     {
18         printf("%s", read_mag[i]);
19     }
20
21     close(readx);
22
23     return 0;
24 }
```

myMore.c - Code::Blocks 20.03

File Edit View Search Project Build Debug Fortran wxSmith Tools Plugins DoryBlocks Settings Help

Management Projects Files FSymbols Workspace

Logs & others

C:\Users\muidg\Desktop\gilem Sistemleri Odev\myMore.c C/C++ Windows (CR+LF) WINDOWS-1254 Line 1, Col 1, Pos 0 Insert Read/Write default