T.C

HASAN FERDİ TURGUTLU TEKNOLOJİ FAKÜLTESİ

YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ

YZM 3102 – İŞLETİM SİSTEMLERİ FİNAL ÖDEVİ

HAZIRLAYANLAR:

162803034 Sefer MİRZA (II. Öğretim)

172802031 İrem ÇELİKBİLEK (I. Öğretim)

172802033 Kadir Can KARADEMİR (I. Öğretim)

172803016 Bünyamin KÜÇÜK (II. Öğretim)

Danışman:

Dr. Öğr. Üyesi Mansur Alp TOÇOĞLU

MANİSA 2020

İçindekiler

1) Projenin Amacı:

myData (parent) ve myMore (child) prosesleri arasındaki haberleşmeyi sağlamak. Bu haberleşmede; myData write, myMore read işlemini gerçekleştirecektir.

2) Projenin Kapsamı:

Prosesler arasındaki bu haberleşmeyi sağlamak için ordinary pipe kullanılacaktır. Çünkü haberleşme tek yönlü olarak (myData read ve myMore write işlemlerini yapacak şekilde) gerçekleşecektir. myData prosesi kendi başına da çalışabilecektir ancak myMore prosesi kendi başına çalışamayacak myData ile birlikte çalışacaktır.

3) Kodlar ve Açıklamaları:

3.1 myData Prosesi

3.1.1 Gerekli Kütüphaneler:

```
1   #include <sys/wait.h> // wait() Fonksiyonu Kütüphanesi
2   #include <stdlib.h>
3   #include <stdio.h>
4   #include <string.h>
5   #include <unistd.h>
6
7   // getch() Fonksiyonu İçin MacOS veya Linux Kütüphane Seçimi
8   #ifdef __APPLE__
9   #include <termios.h>
10  #else
11  #include <termio.h>
12  #endif
```

3.1.2 Tanımlamalar:

```
14  #define BUFFER_SIZE 500          // Satır çok uzun olduğunda konsol yad
15  #define MAXIMUM_LINE_NUMBER 100 // Dosyadaki maksimum satır sayısı
16  #define LINES_PER_TURN 24        // myMore ile basılacak satır sayısı
17  #define READ_END 0
18  #define WRITE_END 1
```

#define yönergesi kaynak kodumuzdaki makroların tanımlanmasına izin verir. Bu makro tanımları, kodumuz boyunca kullanım için sabit değerlerin bildirilmesine izin verir.

3.1.3 getch () Fonksiyonu:

```
int getch(void)
{
    struct termios oldattr, newattr;
    int ch;
    tcgetattr(STDIN_FILENO, &oldattr);
    newattr = oldattr;
    newattr.c_lflag &= ~(ICANON | ECHO);
    tcsetattr(STDIN_FILENO, TCSANOW, &newattr);
    ch = getchar();
    tcsetattr(STDIN_FILENO, TCSANOW, &oldattr);

    return ch;
}
```

getch () fonksiyonu, kullanıcının devam etmek için girdiği tuşu kontrol eder. Bu sayede proses çalışmaya devam edebilir ya da durdurulabilir.

3.1.4 fileSize () Fonksiyonu:

```
int fileSize(FILE *fp)
{
    fseek(fp, 0L, SEEK_END);
    int fileSize = ftell(fp);
    fseek(fp, 0L, SEEK_SET);

    return fileSize;
}
```

filesize() fonksiyonu, byte olarak dosyanın boyutunu verir.

3.1.5 *getFile () Fonksiyonu:

```
FILE *getFile(char *filePath)
{
    FILE *fp;
    fp = fopen(filePath, "r");

    if (fp != NULL)
        return fp;

    else
    {
        perror("Error");
        exit(1);
    }
}
```

*getFile fonksiyonu, dosyanın işaretçisini verir.

3.1.6 readAllFile () Fonksiyonu:

```c
int readAllFile(FILE *fp)
{
    int byteOfFile = fileSize(fp);
    char *records = malloc(byteOfFile + 1);

    if (!fp)
        return -1;

    while (fgets(records, byteOfFile, fp) != NULL) // Dosyanın sonuna(\0) kadar okur.
        printf("%s", records);

    fclose(fp);

    return 0;
}
```

readAllFile () fonksiyonu, dosyayı okurken aynı zamanda dosyayı ekrana bastırır.

3.1.7 Main Fonksiyonu:

```c
int main(int argc, char *argv[])
{
    FILE *fp = getFile(argv[1]);

    // Eğer girdi "./myData <filename>" ise bütün dosyayı okur.
    if (argc == 2)
    {
        readAllFile(fp);
    }
    // Eğer girdi "./myData <filename> = myMore" ise dosyayı 24 satırlık parçalar halinde okur.
    else if (!strcmp(argv[2], "=") && !strcmp(argv[3], "myMore") && argc == 4)
    {
        char buffer[BUFFER_SIZE];
        char inputChar;                                  // Bu değişken girdileri tutar.
        int isFinished = 0;                              // Bu değişken bitip bitmediğini tutar.
        char write_msg[MAXIMUM_LINE_NUMBER][BUFFER_SIZE]; // Bu değişken pipe aracılığıyla gönderilen değişkeni tutar.
```

Öncelikle myData prosesinin tek başına mı yoksa myMore prosesi ile beraber mi çalıştığının kontrolünü yapıyoruz. Eğer myData tek başına çalışıyorsa bütün dosyayı okuyoruz, eğer myMore ile birlikte çalışırsa belirlediğimiz buffer_size ve max satır sayısına kadar (24 satır) okuyoruz.

```
95          do
96          {
97              pid_t pid;
98              int fd[2];
99              pipe(fd);
100             pid = fork(); // Program çatallandırıldı.
101
102             // Çatallandırma başarısız olursa buraya girecek.
103             if (pid < 0)
104             {
105                 fprintf(stderr, "Çatallandırma başarısız oldu.");
106                 return 1;
107             }
```

Sonra do while döngüsünün içine giriyoruz. Bu döngü yazma işlemi bitene kadar veya kullanıcının q tuşuna basması durumuna kadar devam edecektir. İki prosesimizin haberleşebilmesi için pipe oluşturuyoruz ve myData prosesimizi forkluyoruz. Eğer forklama başarısız olursa ekrana mesaj veriyoruz.

```
// Child Process
if (pid == 0)
{
    close(fd[WRITE_END]);

    char read_end_pipe[10]; // Bu değişken pipe'ın okuma ucunu tutar.
    sprintf(read_end_pipe, "%d", fd[READ_END]);

    char *arguments[3] = {argv[3], read_end_pipe, NULL}; // Bu değişken myMore programına gönderilecek olan parametrelerin dizisini tutar.

    // Execv fonksiyonu ile myMore'u çalıştırırız.
    if (execv(argv[3], arguments) == -1)
    {
        perror("myMore programı çalıştırılamadı.");
        return -1;
    }
}
```

Forklama işlemi sonrası eğer pid 0 ise bu bizim child prosesimizdir. Child proses okuma işlemi yapabildiğinden dolayı yazma işlemini sonlandırıyoruz. Execv () fonksiyonu ile de myMore fonksiyonumuzu çalıştırıyoruz. Eğer çalıştıramazsak da hata mesajı gönderiyoruz.

```c
// Parent Process
if (pid > 0)
{
    int lineCount = 0; // Bu değişken kaç satır okunduğunu tutar.

    //24 satır okur eğer 24 satırdan daha az kaldıysa tamamını okur.
    for (int i = 0; i < LINES_PER_TURN; ++i)
    {
        // Satırları okur
        if (fgets(buffer, sizeof(buffer), fp) > 0)
        {
            strcpy(write_msg[i], buffer);
            lineCount++;
        }
        // Dosya okuma bittiyse döngüden çıkar
        else
        {
            isFinished = 1;
            break;
        }
    }

    close(fd[READ_END]);
    write(fd[WRITE_END], &write_msg, lineCount * BUFFER_SIZE);
    close(fd[WRITE_END]);

    wait(NULL);
}
```

Forklama işlemi sonrası pid değerimiz 0'dan büyükse bu bizim parent prosesimizdir. Parent prosesi yazma işlemi yaptığından dolayı text'deki değerleri 24'er ve eğer 24'ten az ise kalan satır kadar okuyarak yazma işlemi yapar. Yazma işlemi bittikten sonra da yazma işlemine son verilir ve child prosesin işini tamamlaması beklenir.

```
            if (isFinished == 0)
                printf("\n ******************** DEVAM ETMEK İÇİN SPACE TUŞUNA BASIN ********************\n\n");
            else
            {
                printf("\n ******************** DOSYA TAMAMEN OKUNDU ********************\n\n");
                return 0;
            }

            inputChar = getch(); //Kullanıcının girdisi

        } while (isFinished != 1 && inputChar != 'q');
    }
    else
    {
        printf("\nKullanım Hatası!\nDosyanın tamamını okumak için:\t\t\"./myData <Metin Dosyası Yolu>\"\n
        Dosyayı satır satır okumak için:\t\"./myData <Metin Dosyası Yolu> = myMore\"\n");
        return 1;
    }

    return 0;
}
```

Do-While döngüsünün sonunda kullanıcıdan girdi bekliyoruz. Alınan girdiye göre prosesi devam ettiriyor ya da sonlandırıyoruz ve bu bilgiyi ekranda gösteriyoruz.

3.2 myMore Prosesi:

3.2.1 Gerekli Kütüphaneler:

```
1   #include "stdlib.h"
2   #include "stdio.h"
3   #include "unistd.h"
```

3.2.2 Tanımlamalar:

```
5   #define BUFFER_SIZE 500
6   #define MAXIMUM_LINE_NUMBER 100
7   #define LINES_PER_TURN 24
8
```

3.2.3 Main Fonksiyonu:

```
9   int main(int argc, char *argv[])
10  {
11      int read_exit = atoi(argv[1]);
12      char read_msg[MAXIMUM_LINE_NUMBER][BUFFER_SIZE];
13
14      read(read_exit, &read_msg, LINES_PER_TURN * BUFFER_SIZE);
15
16      for (int i = 0; i < LINES_PER_TURN; ++i)
17      {
18
19          printf("%s", read_msg[i]);
20      }
21
22      close(read_exit);
23      return 0;
24  }
```

Main fonksiyonu, myData'daki verileri okuma işlemini gerçekleştirir. (24'er 24'er olacak şekilde) Okuma işlemini gerçekleştirdikten sonra da okuma işlemini sonlandırıyoruz.