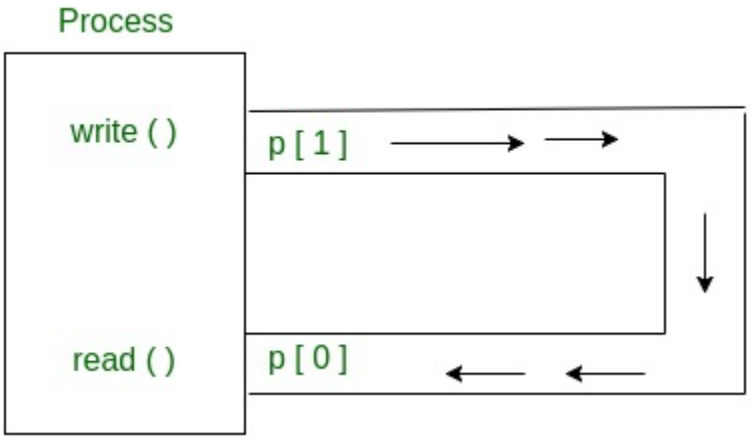
Group Project 1

Warm Up for Interprocess Communication

2/25/2024

CECS 326 - Operating Systems



Brandon Quach (#029133232)

Michael Kim (#025633788)

Project demo:

https://youtu.be/ggRGpm7Mtc4

Project Description:

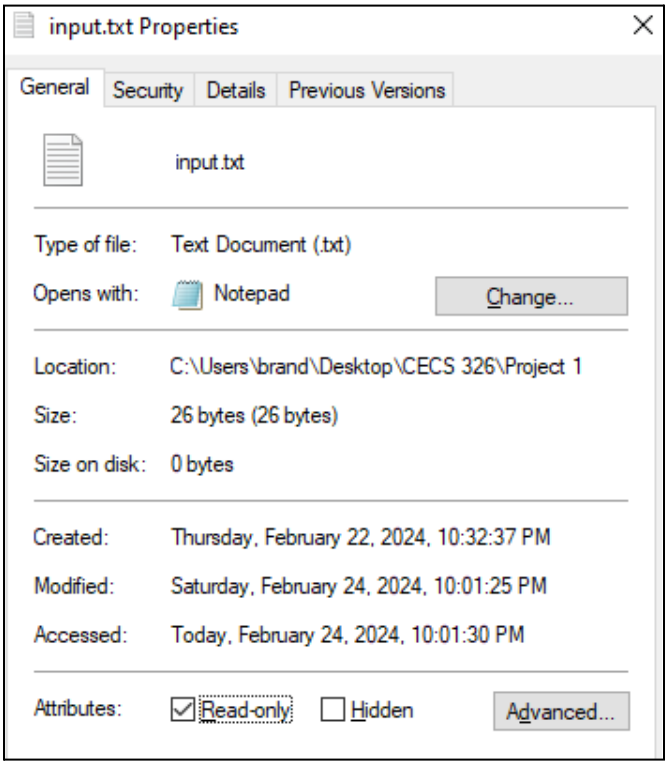
For this project, we created and ran a program named filecopy.c which uses pipes to communicate across processes. The program requires 2 parameters: the name of the source and destination files. A pipe will be created which will get the contents of the source file and transfer it to a pipe. A child process will be used to read from the pipe and write the contents to the destination file.

The filecopy.c program shows how pipes can be used to transport data across 2 different processes efficiently. Through the use of pipes, the application removes the need to temporarily store the contents of the file in the memory. This opens up the opportunity to copy files with very little memory use.

Steps:

1. Download the filecopy.c program and run the code
    a. If you are using Windows, a Linux terminal will be needed to run the program
2. Type in *gcc filecopy.c -o filecopy*
3. Command to copy the source file is *./filecopy source_file destination_file*
4. Command to read what's inside the txt file is *cat file_name*
5. If there is an error, the program will let you know what the problem is

Code analysis:



Source_file in read-only mode

```
int file[2];
pid_t childID;
char readBuffer[BUFFER_SIZE];
pipe(file);
```

Pipe creation

```
//Fork a Child Process
childID = fork();
```

fork() system call to create a child process

```
if (childID == 0)
{
    close(file[1]);
    while (read(file[0], readBuffer, sizeof(readBuffer)) > 0)
    {
        write(destFile, readBuffer, strlen(readBuffer));
    }
    //close files
    close(file[0]);
    close(destFile);
}
```

Parent process: reads from the source file and writes to the end of the pipe

```
//Write to destination file
else
{
    close(file[0]);
    while (read(sourceFile, readBuffer, sizeof(readBuffer)) > 0)
    {
        write(file[1], readBuffer, sizeof(readBuffer));
        memset(readBuffer, 0, BUFFER_SIZE);
    }
    close(file[1]);
    close(sourceFile);
    wait(NULL);
    //File copy successfully status message with file name
    printf("File successfully copied from \'%s",filename1);
    printf("\' to \'%s",filename2);
    printf("\'");
}
```

Child process: reads from the end of the pipe and writes to the destination file

```
//close files
close(file[0]);
close(destFile);
```

Close file descriptors

```c
//Error when user inputs more than 2 parameters
if( argc != 3 )
{
    printf("ERROR: Need exactly 2 parameters.\n");
    exit(1);
}
```

```c
//Error when the source file doesn't exist
if (sourceFile == -1 || destFile == -1)
{
    printf("Error: Unable to open source file \'%s",filename1);
    printf("\'");
    exit(1);
}
```

Error handling

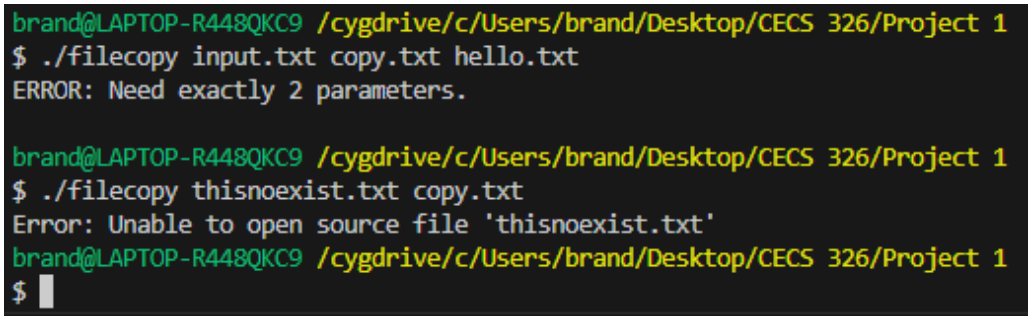
Program output:

```
brand@LAPTOP-R448QKC9 /cygdrive/c/Users/brand/Desktop/CECS 326/Project 1
$ gcc filecopy.c -o filecopy

brand@LAPTOP-R448QKC9 /cygdrive/c/Users/brand/Desktop/CECS 326/Project 1
$ cat input.txt
hello, this program works!
brand@LAPTOP-R448QKC9 /cygdrive/c/Users/brand/Desktop/CECS 326/Project 1
$ cat copy.txt

brand@LAPTOP-R448QKC9 /cygdrive/c/Users/brand/Desktop/CECS 326/Project 1
$ ./filecopy input.txt copy.txt
File successfully copied from 'input.txt' to 'copy.txt'
brand@LAPTOP-R448QKC9 /cygdrive/c/Users/brand/Desktop/CECS 326/Project 1
$ cat copy.txt
hello, this program works!
brand@LAPTOP-R448QKC9 /cygdrive/c/Users/brand/Desktop/CECS 326/Project 1
$ cat input.txt
hello, this program works!
brand@LAPTOP-R448QKC9 /cygdrive/c/Users/brand/Desktop/CECS 326/Project 1
$
```

Snippet showing program working



Snippet showing error handling

What each individual did:

Brandon: Brandon took the role in the completion of the lab report and ensuring that all project deliverables were included. In addition to his lab report responsibilities, he also worked on the code with Michael and input the proper comments to show the design.

Michael: Michael took a technical role within the group, with a primary focus on both software. His responsibilities encompassed software development, creation, and software components. In addition to his software responsponsibilities, he also focused on the video.

Work split was mostly 50/50 when it came to the software and its deliverables.

References:

https://www.gnu.org/software/libc/manual/html_node/Creating-a-Pipe.html

https://www.youtube.com/watch?v=8Q9CPWuRC6o