**MULTIPLE FILE**

**COPYING**

**LAB # 0****7**

**Fall 2023**

**CSE-302L**

**Systems Programming Lab**

Submitted by: **AIMAL KHAN**

Registration No.: **21PWCSE1996**

Class Section: **A**

“On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work.”



Student Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Submitted to:

**Engr. Abdullah Hamid**

Sunday, January 28, 2024

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

**CSE 302L: SYSTEMS PROGRAMMING LAB**

**LAB ASSESSMENT RUBRICS**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Criteria & Point Assigned** | **Outstanding 2** | **Acceptable 1.5** | **Considerable 1** | **Below Expectations 0.5** | **Score** |
| **Attendance and**  **Attentiveness in Lab**  PLO08 | Attended in proper Time and attentive in Lab | Attended in proper Time but not attentive in Lab | Attended late but attentive in Lab | Attended late not attentive in Lab |  |
| **Capability of writing Program/Algorithm/Drawing Flow Chart**  PLO1, PLO2, PLO3, PLO5 | Right attempt/ no errors and well formatted | Right attempt/ no errors but not well formatted | Right attempt/ minor errors and not well formatted | Wrong attempt |  |
| **Result or Output/ Completion of target in Lab**  PLO9 | 100% target has been completed and well formatted. | 75% target has been completed and well formatted. | 50% target has been completed but not well formatted. | None of the outputs are correct. |  |
| **Overall, Knowledge**  PLO10, | Demonstrates excellent knowledge of lab | Demonstrates good knowledge of lab | Has partial idea about the Lab and procedure followed | Has poor idea about the Lab and procedure followed |  |
| **Attention to Lab Report**  PLO4, | Submission of Lab Report in Proper Time i.e., in next day of lab, with proper documentation. | Submission of Lab Report in proper time but not with proper documentation. | Late Submission with proper documentation. | Late Submission very poor documentation |  |

**Instructor:**

|  |  |
| --- | --- |
| Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  |  |
|  |  |

**Multiple File Copying**

Objectives:

Learn about these concepts

* Handling of I/O from multiple sources
* Monitoring multiple files

Tasks:

**Task 1**: Write a program that copies two files sequentially in a single process.

**Code in C:**

#*include* <stdlib.h>

#*include* <stdio.h>

#*include* <unistd.h>

#*include* "../../reusable\_code\_snippets/readWrite.h"

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

{

*if* (argc < 3)

    {

*fprintf*(*stderr*, "Usage: %s FILE\_1 FILE\_2 [FILE\_N...]\n", argv[0]);

*return* 1;

    }

*if* (argc % 2 == 0)

    {

*perror*("Argument must be even in count.\n");

*return* 1;

    }

    // *int readWriteReturnValue;*

*for* (int i = 1, readWriteReturnValue; i < argc; i += 2)

    {

        readWriteReturnValue = *readWrite*(argv[i], argv[i + 1]);

*if* (readWriteReturnValue < 0)

        {

*perror*("Something went wrong while reading or writing a file.\n");

*return* 1;

        }

*printf*("Copied from SRC: %s to DIST: %s.\n", argv[i], argv[i + 1]);

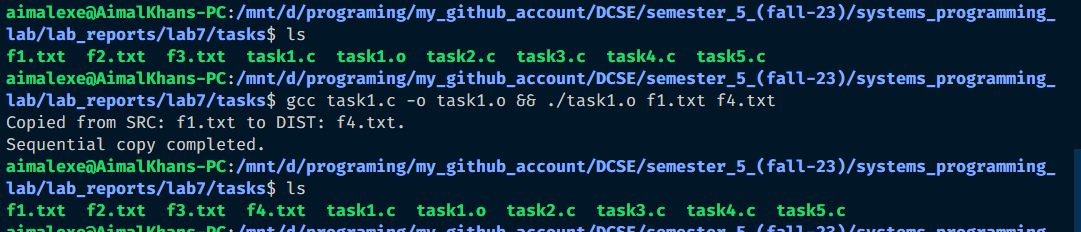
    }

*printf*("Sequential copy completed.\n");

*return* 0;

}

**Output:**

****

**Task 2**: Write a program that monitors two files using 'select'.

**Code in C:**

// *Task 2: Write a program that monitors two files using 'select'.*

#*include* <stdlib.h>

#*include* <stdio.h>

#*include* <unistd.h>

#*include* <sys/select.h>

#*include* "../../reusable\_code\_snippets/readWrite.h"

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

{

*if* (argc != 3)

    {

*fprintf*(*stderr*, "Usage: %s FILE\_1 FILE\_2\n", argv[0]);

*return* 1;

    }

    // *1. Opening the two files.*

    int file1 = *open*(argv[1], *O\_RDWR*);

*if* (file1 == -1)

    {

*fprintf*(*stderr*, "Something went wrong while opening the file: %s due to %s\n", argv[1], *strerror*(*errno*));

*return* 1;

    }

    int file2 = *open*(argv[2], *O\_RDWR*);

*if* (file2 == -1)

    {

*fprintf*(*stderr*, "Something went wrong while opening the file: %s due to %s\n", argv[2], *strerror*(*errno*));

*return* 1;

    }

    // *2. Collecting arguments for select function.*

    int maxFd = file1 >= file2 ? file1 : file2;

    fd\_set readset;

*FD\_ZERO*(&readset);

*FD\_SET*(file1, &readset);

*FD\_SET*(file2, &readset);

    // *calling the function select.*

    int readyFile = *select*(maxFd + 1, &readset, *NULL*, *NULL*, *NULL*);

    // *Checking for ready file(S):*

*if* (*FD\_ISSET*(file1, &readset))

*printf*("The file: %s with FD = %d is ready.\n", argv[1], file1);

*if* (*FD\_ISSET*(file2, &readset))

*printf*("The file: %s with FD = %d is ready.\n", argv[2], file2);

    // *Closing the files:*

*if* (*close*(file1) == -1)

    {

*fprintf*(*stderr*, "Something went wrong while closing the file: %s due to %s\n", argv[1], *strerror*(*errno*));

*return* 1;

    }

*if* (*close*(file2) == -1)

    {

*fprintf*(*stderr*, "Something went wrong while closing the file: %s due to %s\n", argv[2], *strerror*(*errno*));

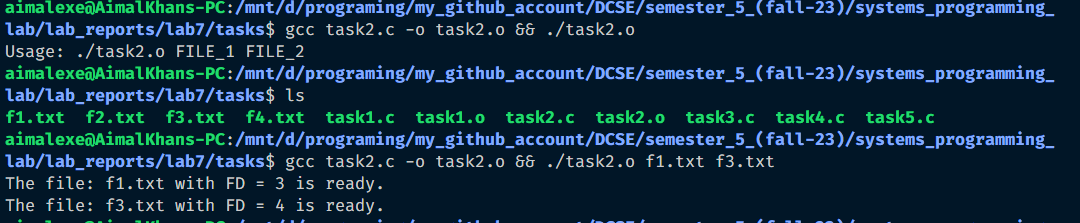
*return* 1;

    }

*return* 0;

}

**Output:**

****

**Task 3** : Write a program that monitors N files using 'select'

**Code in C:**

// *Task 3: Write a program that monitors N files using 'select'.*

#*include* <stdlib.h>

#*include* <stdio.h>

#*include* <unistd.h>

#*include* <errno.h>

#*include* <fcntl.h>

#*include* <string.h>

#*include* <sys/select.h>

#*include* <time.h>

#*include* "../../reusable\_code\_snippets/findMax.h"

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

{

*if* (argc < 2)

    {

*fprintf*(stderr, "Usage: %s FILE\_1 [FILE\_N...]\n", argv[0]);

*return* 1;

    }

    int length = argc - 1;

    int fds[length];

*for* (int i = 0; i < length; i++)

    {

        fds[i] = *open*(argv[i + 1], O\_RDWR);

*if* (fds[i] == -1)

        {

*fprintf*(stderr, "Something went wrong while opening the file: %s due to %s\n", argv[i + 1], *strerror*(errno));

*return* 1;

        }

    }

    fd\_set readSet;

*FD\_ZERO*(&readSet);

    int max = *findMax*(fds, &length);

*for* (int i = 0; i < length; i++)

*FD\_SET*(fds[i], &readSet);

    int ret = *select*(max + 1, &readSet, NULL, NULL, NULL);

*for* (int i = 0; i < length; i++)

*if* (*FD\_ISSET*(fds[i], &readSet))

*printf*("The file: %s with FD = %d is ready.\n", argv[i + 1], fds[i]);

*for* (int i = 0; i < length; i++)

*if* (*close*(fds[i]) == -1)

        {

*fprintf*(stderr, "Something went wrong while closing the file: %s due to %s\n", argv[i + 1], *strerror*(errno));

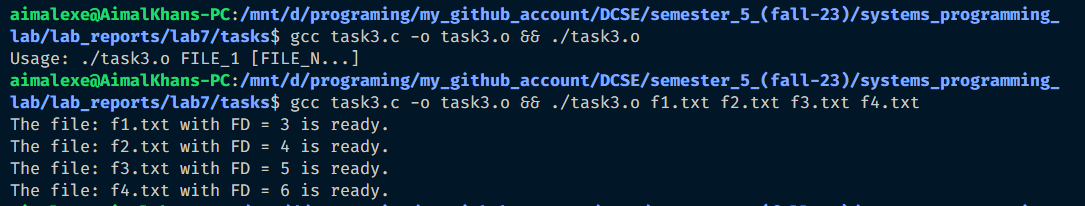
*return* 1;

        }

*return* 0;

}

**Output:**

****

**Task 4**: Write a program that creates two child processes, both child read from the same source and writes to two separate destination, make sure you use select to monitor if the source is ready to be read from.

**Code in C:**

/\*

*Task 4: Write a program that creates two child processes, both child read from the*

*same source and writes to two separate destination, make sure you use select to*

*monitor if the source is ready to be read from.*

\*/

#*include* <stdlib.h>

#*include* <stdio.h>

#*include* <unistd.h>

#*include* <sys/types.h>

#*include* <sys/select.h>

#*include* <sys/wait.h>

#*include* "../../reusable\_code\_snippets/readWrite.h"

#*include* "../../reusable\_code\_snippets/findMax.h"

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

{

*if* (argc <= 3)

    {

*fprintf*(*stderr*, "Usage:\n%s READ\_FROM\_FILE\_1 WRITE\_TO\_FILE\_1 WRITE\_TO\_FILE\_2 [WRITE\_TO\_FILE\_N...]\n", argv[0]);

*return* 1;

    }

    int readFd = *open*(argv[1], *O\_RDONLY*);

*if* (readFd == -1)

    {

*fprintf*(*stderr*, "Cannot open source file %s: %s\n", argv[1], *strerror*(*errno*));

*return* 1;

    }

*for* (int i = 2; i < argc; i++)

    {

        pid\_t pid = *fork*();

*if* (pid == -1)

        {

*perror*("Failed to create child process.\n");

*return* 1;

        }

*else* *if* (pid == 0)

        {// *Child*

            int writeFd = *open*(argv[i], *O\_WRONLY* | *O\_TRUNC* | *O\_CREAT*, *S\_IRWXU* | *S\_IRGRP* | *S\_IROTH*);

*if* (writeFd == -1)

            {

*fprintf*(*stderr*, "Cannot open destination file %s: %s\n", argv[i], *strerror*(*errno*));

*return* 1;

            }

            fd\_set readSet;

*FD\_ZERO*(&readSet);

*FD\_SET*(readFd, &readSet);

            // *Child process has to select on the readFd again because file descriptors are shared.*

*if* (*select*(readFd + 1, &readSet, *NULL*, *NULL*, *NULL*) > 0)

            {

*if* (*FD\_ISSET*(readFd, &readSet))

*if* (*readWriteOnly*(&readFd, &writeFd) == -1)

                    {

*fprintf*(*stderr*, "Error reading/writing files.\n");

*return* 1;

                    }

            }

*else*

            {

*perror*("Error while monitoring files.\n");

*return* 1;

            }

*if* (*close*(writeFd) == -1)

            {

*fprintf*(*stderr*, "Something went wrong while closing the file: %s due to %s\n", argv[i], *strerror*(*errno*));

*return* 1;

            }

*printf*("Process -> %d, %d.\n", i, writeFd);

*return* 0;// *Child process exits after writing.*

        }

    }

    // *Parent process*

*for* (int i = 2; i < argc; i++)

*wait*(*NULL*);// *Wait for all children to finish.*

    // *Close the source file descriptor in the parent as well.*

*if* (*close*(readFd) == -1)

    {

*fprintf*(*stderr*, "Something went wrong while closing the file: %s due to %s\n", argv[1], *strerror*(*errno*));

*return* 1;

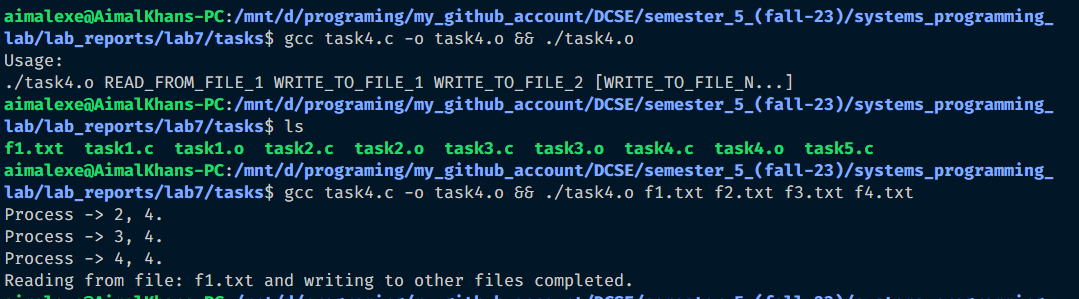
    };

*printf*("Reading from file: %s and writing to other files completed.\n", argv[1]);

*return* 0;

}

**Output:**

****

**Task 5:** Write a function that creates a delay of N seconds using select function. Pass N as an argument to the function

**Code in C:**

/\*

*Task 5: Write a function that creates a delay of N seconds using select function. Pass*

*N as an argument to the function*

\*/

#*include* <stdlib.h>

#*include* <stdio.h>

#*include* <unistd.h>

#*include* <errno.h>

#*include* <string.h>

#*include* <sys/select.h>

#*include* <time.h>

int *doDelay*(int sec, int microSec)

{

    struct timeval t;

    t.tv\_sec = sec;

    t.tv\_usec = microSec;

    int ret = *select*(1, *NULL*, *NULL*, *NULL*, &t);

*return* 0;

}

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

{

*if* (argc != 3)

    {

*fprintf*(*stderr*, "Usage: %s <seconds> <micro\_seconds>.\n", argv[0]);

*return* 1;

    }

*printf*("Starting program...\n");

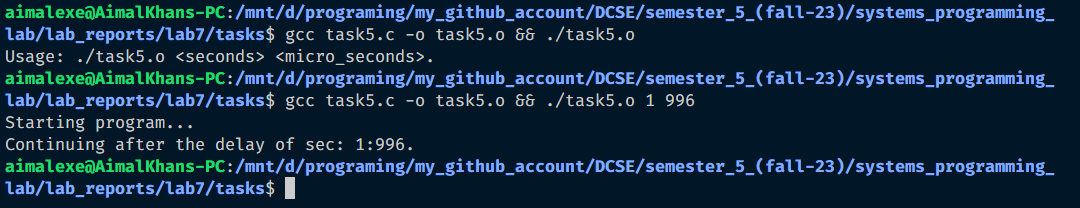
    int ret = *doDelay*(*atoi*(argv[1]), *atoi*(argv[2]));

*printf*("Continuing after the delay of sec: %d:%d.\n", *atoi*(argv[1]), *atoi*(argv[2]));

*return* 0;

}

**Output:**

****

Reference:

To view my codes, please refer to my GitHub account:  [https://github.com/aimalexe/DCSE/tree/main/semester\_5\_(fall-23)/systems\_programming\_lab/lab\_reports](%20https://github.com/aimalexe/DCSE/tree/main/semester_5_(fall-23)/systems_programming_lab/lab_reports) .

Conclusion:

In this lab I have learned different techniques about monitoring file like using multiple processes, a system call named *select* which monitor files for reading, writing and errors. It can also be used to stop the program for a while. Now I am able to manage my file monitoring with the findings.

The End.