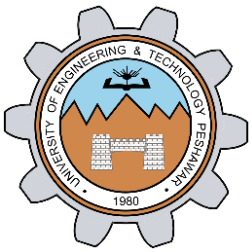
SYSTEM PROGRAMMING

ASSIGNMENT



Fall 2023

CSE-302

System Programming

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:

Dr. Madiha Sher

Saturday, January 20, 2024

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

Question 1: *Write a program that searches for a file passed to it as a command line argument in all the provided paths. Take paths as CLA.*

***Sample Run**: ./find . .. ~/Desktop*

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <dirent.h>
#include <sys/stat.h>
#include <string.h>

void searchFile(const char *target, const char *path)
{
    struct stat entryStatistics;
    struct dirent *directoryEntry;
    DIR *dp;

    // open path(directory)
    if ((dp = opendir(path)) == NULL)
    {
        perror("Error while opening directory.\n");
        return;
    }

    // read the directory entries one by one
    while ((directoryEntry = readdir(dp)) != NULL)
    {
        char newPath[1024];

        // skip . and ..
        if ((!strcmp(directoryEntry->d_name, ".")) ||
(!strcmp(directoryEntry->d_name, "..")))
            continue;

        // S_ISDIR(entryStatistics.st_mode) ? printf("%s:\n",
directoryEntry->d_name) : printf("%s\n", directoryEntry->d_name);
        // update the path by appending the file name with it.
        snprintf(newPath, sizeof(newPath), "%s/%s", path,
directoryEntry->d_name);

        if (stat(newPath, &entryStatistics) == -1)
        {
            perror("Error while traversing statistics.\n");
            return;
        }

        if (S_ISDIR(entryStatistics.st_mode)) // if directory search
for file in subdirectories.
            searchFile(target, newPath);
        else if (S_ISREG(entryStatistics.st_mode)) // if regular
file compare the name of entry and target.
```

```
            if (!strcmp(target, directoryEntry->d_name))
                printf("%s/%s\n", path, directoryEntry->d_name);
    }

    if (closedir(dp) == -1)
    {
        perror("Error while opening directory.\n");
        return;
    }

    return;
}

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

    if (argc < 3)
    {
        fprintf(stderr, "Need atleast two arguments. Usage:\n%s
FILE_NAME DIRECTORY_PATH [DIRECTORY_PATH ...]\n", argv[0]);
        return 1;
    }

    for (int i = 2; i < argc; i++)
    {
        printf("Searching the file '%s' in directory '%s'\n",
argv[1], argv[i]);
        searchFile(argv[1], argv[i]);
    }

    return 0;
}
```

Output:

```
aimalexe@AimalKhans-PC:/mnt/d/computer_system_course/semester_5_(fall-23)/systems_programming/assignments
$ gcc question1.c -o question1.o && ./question1.o f0.txt .. ../..
Searching the file 'f0.txt' in directory '..'
../assignments/f0.txt
Searching the file 'f0.txt' in directory '../..'
../../systems_programming/assignments/f0.txt
```

Question 2: *Write a program to implement ls command. Take the name of the directory to be listed from command line. Also print the path of CWD.*

***Sample Run**: ./t1.o SP*

Code:

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <dirent.h>
#include <sys/stat.h>
```

```
#include <time.h>
#include <grp.h>
#include <pwd.h>

int displayFileStatistics(struct dirent *directoryEntry)
{
    struct stat entryStatistics;
    struct group *g;
    struct passwd *p;
    char *ctime_no_newline;

    if (stat(directoryEntry->d_name, &entryStatistics) == -1)
    {
        perror("Error in statistics of the directory contents.\n");
        return -1;
    }

    if (!strcmp(directoryEntry->d_name, ".") ||
!strcmp(directoryEntry->d_name, ".."))
        return 0;

    printf("\t");
    // print if entry is a directory or a file
    S_ISDIR(entryStatistics.st_mode) ? printf("d ") : printf("- ");

    // printing permissions by taking bitwise and of permission bit
with mode.
    S_IRUSR & entryStatistics.st_mode ? printf("r") : printf("-");
    S_IWUSR & entryStatistics.st_mode ? printf("w") : printf("-");
    S_IXUSR & entryStatistics.st_mode ? printf("x") : printf("-");
    S_IRGRP & entryStatistics.st_mode ? printf("r") : printf("-");
    S_IWGRP & entryStatistics.st_mode ? printf("w") : printf("-");
    S_IXGRP & entryStatistics.st_mode ? printf("x") : printf("-");
    S_IROTH & entryStatistics.st_mode ? printf("r") : printf("-");
    S_IWOTH & entryStatistics.st_mode ? printf("w") : printf("-");
    S_IXOTH & entryStatistics.st_mode ? printf("x") : printf("-");

    // No of links pointing to file or directory
    printf(" %ld", entryStatistics.st_nlink);

    // User Name from uid
    p = getpwuid(entryStatistics.st_uid);
    printf(" %s", p->pw_name);

    // Group Name from gid
    g = getgrgid(entryStatistics.st_gid);
    printf(" %s", g->gr_name);

    // Time and date of last access
    ctime_no_newline = strtok(ctime(&entryStatistics.st_ctime),
"\n");
    printf(" %s", ctime_no_newline);

    // Entry name
```

```
    printf(" %s\n", directoryEntry->d_name);
    return 0;
}

int main(int argc, char *argv[])
{
    if (argc > 3)
    {
        fprintf(stderr, "Need at most three args. Usage:\n%s
[OPTIONAL] [-l | FILE_NAME]\n", argv[0]);
        return 1;
    }

    DIR *thisDirectory = opendir(".");
    struct dirent *directoryEntries;

    // ls
    if (argc == 1)
    {
        printf("Directory Content:\n");
        while ((directoryEntries = readdir(thisDirectory)) != NULL)
        {
            if (!strcmp(directoryEntries->d_name, ".") ||
!strcmp(directoryEntries->d_name, ".."))
                continue;
            printf("\t%s\n", directoryEntries->d_name);
        }
    }

    // ls -l
    else if ((argc == 2) && (!strcmp(argv[1], "-l")))
    {
        printf("Directory Content Statistics:\n");
        while ((directoryEntries = readdir(thisDirectory)) != NULL)
            if (displayFileStatistics(directoryEntries) == -1)
                return 1;
    }

    // ls file.xyz
    else if (argc == 2)
    {
        while ((directoryEntries = readdir(thisDirectory)) != NULL)
            if (!strcmp(directoryEntries->d_name, argv[1]))
            {
                printf("File Found:\n\t%s\n", directoryEntries-
>d_name);
                break;
            }
    }

    // ls -l file.xyz
    else if ((argc == 3) && (!strcmp(argv[1], "-l")))
    {
        while ((directoryEntries = readdir(thisDirectory)) != NULL)
```

```
            if (!strcmp(argv[2], directoryEntries->d_name))
            {
                printf("File Found. Statistics:\n");
                if (displayFileStatistics(directoryEntries) == -1)
                    return 1;
                break;
            }
    }

    return 0;
}
```

Output:

```
aimalexe@AimalKhans-PC:/mnt/d/computer_system_course/semester_5_(fall-23)/systems_programming/assignments
$ gcc question2.c -o question2.o && ./question2.o -l
Directory Content Statistics:
        - rwxrwxrwx 1 aimalexe aimalexe Sat Jan 20 00:58:08 2024 f0.txt
        - rwxrwxrwx 1 aimalexe aimalexe Sat Jan 20 00:38:43 2024 f1.txt
        - rwxrwxrwx 1 aimalexe aimalexe Sat Jan 20 01:01:36 2024 f3.txt
        - rwxrwxrwx 1 aimalexe aimalexe Sat Jan 20 01:02:03 2024 f4.txt
        - rwxrwxrwx 1 aimalexe aimalexe Fri Jan 19 21:14:31 2024 question1.c
        - rwxrwxrwx 1 aimalexe aimalexe Thu Jan 25 03:09:45 2024 question1.o
        - rwxrwxrwx 1 aimalexe aimalexe Sat Jan 20 00:03:02 2024 question10.c
        - rwxrwxrwx 1 aimalexe aimalexe Sat Jan 20 00:02:01 2024 question10.o
        - rwxrwxrwx 1 aimalexe aimalexe Sat Jan 20 01:06:42 2024 question11.c
        - rwxrwxrwx 1 aimalexe aimalexe Sat Jan 20 01:02:03 2024 question11.o
        - rwxrwxrwx 1 aimalexe aimalexe Fri Jan 19 21:26:43 2024 question2.c
        - rwxrwxrwx 1 aimalexe aimalexe Thu Jan 25 03:16:09 2024 question2.o
```

Question 3: *Write a program that finds a file in a directory. Program shall receive the name of the file & directory from command line.*

***Sample Run**: ./find.o SP task1.c*

*I*mplemented in Question # 1. Refer to that same code and output.

Question 4: *Write a program that implements FTP Server. Client requests for the contents of a specific directory. Server responds with the list of files/directories*.

Client Code:

```
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <stdlib.h>
#include <errno.h>
#include <sys/select.h>

#define BUFF_SIZE 128

int main(int argc, char *argv[])
{
    if (argc != 2)
    {
```

```
        fprintf(stderr, "Need one arguments. Usage:\n%s
DIRECTORY_PATH\n", argv[0]);
        return 1;
    }
    int fifo = mkfifo("fifo", S_IRWXU);
    int fd = open("fifo", O_RDWR, S_IRWXU);
    if ((fifo == -1) && (errno != EEXIST))
    {
        perror("Error: While creating the fifo.\n");
        return 1;
    }
    char buff[BUFF_SIZE];
    strcpy(buff, argv[1]);
    int wd = write(fd, buff, strlen(buff));
    sleep(4);
    fd_set readSet;
    FD_ZERO(&readSet);
    FD_SET(fd, &readSet);
    int nrfd = select(fd + 1, &readSet, NULL, NULL, NULL);
    while (1)
    {
        FD_ZERO(&readSet);
        FD_SET(fd, &readSet);
        if (FD_ISSET(fd, &readSet))
        {
            for (;;)
            {
                int rd = read(fd, buff, BUFF_SIZE);
                if (rd == 0)
                    break;
                int wr = write(STDOUT_FILENO, buff, rd);
            }
            break;
        }
    }
}
```

Server Code:

```
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <stdlib.h>
#include <errno.h>
#include <sys/select.h>
#include <dirent.h>

#define BUFF_SIZE 128

int main()
{
    int mkffo = mkfifo("fifo", S_IRWXU);
```

```
    int fd = open("fifo", O_RDWR, S_IRWXU);
    if ((fifo == -1) && (errno != EEXIST))
    {
        perror("Error: While creating the fifo.\n");
        return 1;
    }
    char buff[BUFF_SIZE];
    int rd = read(fd, buff, BUFF_SIZE);
    struct dirent *entry;
    DIR *folder;
    folder = opendir(buff);
    while ((entry = readdir(folder)))
    {
        write(fd, entry->d_name, strlen(entry->d_name));
        write(fd, "\n", 1);
    }
}
```

Question 5: *Write a program that implements a simple FTP Server. Client requests for a file and server responds with the contents of the file. Client shall receive the contents and display on STD_OUT.*

Server Code:

```
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <stdlib.h>
#include <errno.h>
#include <sys/select.h>
#include <dirent.h>

int main()
{
    int mkffo = mkfifo("fifo", S_IRWXU);
    int fd = open("fifo", O_RDWR, S_IRWXU);
    char buff[20], message[100];
    int rd = read(fd, buff, 20);
    int fd2 = open(buff, O_RDONLY);
    int rd1 = read(fd2, message, 100);
    write(fd, message, rd);
}
```

Client Code:

```
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <fcntl.h>
#include <sys/stat.h>
```

```c
#include <stdlib.h>
#include <errno.h>
#include <sys/select.h>

int main(int argc, char *argv[])
{
    int mkffo = mkfifo("fifo", S_IRWXU);
    int fd = open("fifo", O_RDWR, S_IRWXU);
    if (mkffo == -1 && errno != EEXIST)
        perror("Error: ");
    char buff[20], message[100];
    strcpy(buff, "f0.txt");
    int wd = write(fd, buff, 20);
    sleep(1);
    int rd = read(fd, message, 100);

    int wr = write(1, message, rd);
}
```

Question 6: *Write a program for continuous communication (2-Way) between parent & child process using pipes.*

Code:

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>

#define BUFF_SIZE 256

int main(void)
{
    int fd1[2], fd2[2];
    pid_t pid;
    int again = 1;
    int readBytes, writeBytes;
    char buff[BUFF_SIZE];

    if ((pipe(fd1) == -1) || (pipe(fd2) == -1))
    {
        perror("Error while creating a pipe\n");
        return 1;
    }

    pid = fork();
    if (pid == -1)
    {
        perror("Error while creating a process.\n");
        return 1;
    }
```

```
    while (again)
    {
        if (pid == 0)
        {
            // process 1 (child) should send a message
            printf("\t CHILD-[%ld]\t", (long)getpid());
            readBytes = read(STDIN_FILENO, buff, BUFF_SIZE);
            if (readBytes == -1)
            {
                fprintf(stderr, "C-[%ld]: Error while reading from
STDIN.\n", (long)getpid());
                return 1;
            }
            writeBytes = write(fd1[1], buff, readBytes);
            if (writeBytes == -1)
            {
                fprintf(stderr, "C-[%ld]: Error while writing to
pipe.\n", (long)getpid());
                return 1;
            }

            // It should receive a reply.
            readBytes = read(fd2[0], buff, BUFF_SIZE);
            if (readBytes == -1)
            {
                fprintf(stderr, "C-[%ld]: Error while reading from
pipe.\n", (long)getpid());
                return 1;
            }
            writeBytes = write(STDOUT_FILENO, buff, readBytes);
            if (writeBytes == -1)
            {
                fprintf(stderr, "C-[%ld]: Error while writing to
STDOUT.\n", (long)getpid());
                return 1;
            }

            // should continue?
            printf("Do you want to continue chat? Enter a number!\n
1. YES\n 2. NO\n");
            scanf("%d", &again);
            if (again != 1)
                break;
        }

        else
        {
            // process 2 (parent) should reply to the a message
            // It should receive a reply.
            printf("PARENT-[%ld]\t", (long)getpid());
            readBytes = read(fd1[0], buff, BUFF_SIZE);
            if (readBytes == -1)
            {
```

```
                fprintf(stderr, "P-[%ld]: Error while reading from
pipe.\n", (long)getpid());
                return 1;
            }
            writeBytes = write(STDOUT_FILENO, buff, readBytes);
            if (writeBytes == -1)
            {
                fprintf(stderr, "P-[%ld]: Error while writing to
STDOUT.\n", (long)getpid());
                return 1;
            }

            readBytes = read(STDIN_FILENO, buff, BUFF_SIZE);
            if (readBytes == -1)
            {
                fprintf(stderr, "P-[%ld]: Error while reading from
STDIN.\n", (long)getpid());
                return 1;
            }
            writeBytes = write(fd2[1], buff, readBytes);
            if (writeBytes == -1)
            {
                fprintf(stderr, "C-[%ld]: Error while writing to
pipe.\n", (long)getpid());
                return 1;
            }

            // should continue?
            printf("Do you want to continue chat? Enter a number!\n
1. YES\n 2. NO\n");
            scanf("%d", &again);
            if (again != 1)
                break;
        }
    }

    if (pid != 0)
        wait(NULL);
    printf("Process [%ld] leaved the chat....", (long)getpid());
    return 0;
}
```

Output:

```
aimalexe@AimalKhans-PC:/mnt/d/computer_system_course/semester_5_(fall-2
$ gcc question6.c -o question6.o && ./question6.o
hi
hello
hi
PARENT-[869]    Do you want to continue chat? Enter a number!
 1. YES
hello
 2. NO
         CHILD-[870]    Do you want to continue chat? Enter a number!
 1. YES
 2. NO
1
1
hi
1
```

Question 7: *Write a program for parallel array addition. The program must create 3 child processes and each child should calculate the sum of the one-third (1/3) of array elements. Parent process shall receive the sum calculated by each child, add them to get final sum and then display it. Make sure there are no orphan child processes. You can use pipes, fifos or return value of child processes for Inter Process Communication.*

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <string.h>
#include <time.h>

#define ARRAY_SIZE 99
#define PROCS_COUNT 3
#define BUFF_SIZE 128

void initArray(int *array, const int length)
{
    srand(time(NULL)); // used before rand to generate random
numbers.
    for (int i = 0; i < length; i++)
        array[i] = rand() % 100;
}

int main(void)
{
    int sum = 0, list[ARRAY_SIZE], fd[2], readBytes;
    pid_t pid;
    char buff[BUFF_SIZE];

    if (pipe(fd) == -1)
    {
        perror("Error while creating pipe.\n");
        return 1;
    }

    initArray(list, ARRAY_SIZE);

    for (int i = 0; i < PROCS_COUNT; i++)
    {
        pid = fork();
        if (pid == -1)
        {
            perror("Error while creating the process.\n");
            return 1;
        }
        else if (pid == 0)
        {
            int localSum = 0;
```

```
            int start = i * ARRAY_SIZE / PROCS_COUNT;
            int end = (i + 1) * ARRAY_SIZE / PROCS_COUNT;
            for (int j = start; j < end; j++)
                localSum += list[j];
            printf("\tThe local sum of child %d is: %d\n", i,
localSum);
            // write local sum to pipe.
            sprintf(buff, "%d", localSum);
            if (write(fd[1], buff, strlen(buff)) == -1)
            {
                perror("Error while writing to pipe.\n");
                return 1;
            }
            return 0;
        }
        else
        {
            wait(NULL);

            // read sum from pipe
            readBytes = read(fd[0], buff, BUFF_SIZE);
            if (readBytes == -1)
            {
                perror("Error while reading from pipe.\n");
                return 1;
            }
            buff[readBytes] = '\0';
            sum += atoi(buff);
        }
    }
    printf("The sum of array is: %d\n", sum);

    if ((close(fd[0]) == -1) || (close(fd[1]) == -1))
    {
        perror("Error while closing the pipe.\n");
        return 1;
    }
    return 0;
}
```

Output:

```
aimalexe@AimalKhans-PC:/mnt/d/computer_system_course/semester_5_(fall-23)/systems_programming/assignments
$ gcc question7.c -o question7.o && ./question7.o
        The local sum of child 0 is: 1704
        The local sum of child 1 is: 1553
        The local sum of child 2 is: 1381
The sum of array is: 4638
```

Question 8: *Write a program that creates a child process. Child process shall send "N" **SIGUSR1** or **SIGUSR2** to parent process. Parent process shall count the number of **SIGUSR2** received.*

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <signal.h>
#include <sys/wait.h>

int count = 0;

void signalHandler(int signo)
{
    // printf("%d signal received.\n", signo);
    if (signo == SIGUSR1)
    {
        count++;
    }
}

int main(void)
{
    struct sigaction act;
    act.sa_handler = signalHandler;
    act.sa_flags = 0;

    if ((sigemptyset(&act.sa_mask) == -1) ||
(sigdelset(&act.sa_mask, SIGUSR1) == -1) || (sigdelset(&act.sa_mask,
SIGINT) == -1))
    {
        perror("Error in setting up mask.\n");
        return 1;
    }

    sigaction(SIGUSR1, &act, NULL);

    pid_t pid = fork();
    if (pid == -1)
    {
        perror("Error while creating the process.\n");
        return 1;
    }
    else if (pid == 0)
    {
        for (int i = 1; i < 12; i++)
        {

            kill(getppid(), SIGUSR1);
            sleep(1);
        }

        exit(0);
    }

    wait(NULL);
```

```
    printf("count = %d\n", count);

    return 0;
}
```

Output:

```
aimalexe@AimalKhans-PC:/mnt/d/computer_system_course/semester_5_(fall-23)/systems_programming/assignments
$ gcc question8.c -o question8.o && ./question8.o
count = 1
```

Question 9: *Write a program that creates a child process & waits for the child process to terminate using* ***pause / sigsuspend / sigwait.***

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <signal.h>
#include <sys/wait.h>

void sigchld_handler(int sig) {
    // Handle SIGCHLD signal (child process termination)
    printf("Child process terminated.\n");
}

int main() {
    pid_t child_pid;

    child_pid = fork();

    if (child_pid == 0) {
        // Child process
        printf("Child process: Running...\n");
        sleep(2); // Simulate some work
        printf("Child process: Exiting.\n");
        exit(0);
    } else if (child_pid > 0) {
        // Parent process
        printf("Parent process: Waiting for child to
terminate...\n");

        // Method 1: Using pause()
        printf("\nMethod 1: Using pause()\n");
        pause(); // Suspend until a signal is received

        // Method 2: Using sigsuspend()
        printf("\nMethod 2: Using sigsuspend()\n");
        sigset_t mask;
        sigemptyset(&mask);
        sigaddset(&mask, SIGCHLD);
        sigsuspend(&mask); // Wait for SIGCHLD while blocking other
signals
```

```
        // Method 3: Using sigwait()
        printf("\nMethod 3: Using sigwait()\n");
        sigwait(&mask, &sig); // Wait for SIGCHLD, returning the
signal

        printf("Parent process: Child terminated.\n");
    } else {
        // Fork failed
        perror("fork");
        exit(1);
    }

    return 0;
}
```

Output:

Question 10: *Write a program that creates 2 threads. Thread 1: Find sum of array elements. Thread 2: Searches for a key in array.*

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>

#define ARRAY_SIZE 100
#define THREADS_COUNT 3

int sum = 0;
int list[ARRAY_SIZE];

void initArray(int *array, const int length)
{
    srand(time(NULL)); // used before rand to generate random
numbers.
    for (int i = 0; i < length; i++)
        array[i] = rand() % 100;
}

void *threadSum(void *arg)
{
    int index = *((int *)arg);
    int start = index * ARRAY_SIZE / (THREADS_COUNT - 1);
    int end = start + index * (ARRAY_SIZE / (THREADS_COUNT - 1) -
1);

    int lsum = 0;

    for (int i = start; i < end; i++)
        lsum += list[i];
```

```
    sum += lsum;
}

void *threadSearch(void *arg)
{
    int target = *((int *)arg);
    int *found = malloc(sizeof(int));
    *found = -1;
    for (int i = 0; i < ARRAY_SIZE; i++)
        if (list[i] == target)
        {
            *found = i;
            pthread_exit((void *)found);
        }
    pthread_exit((void *)found);
}

int main(int argc, char *argv[])
{
    pthread_t tid[THREADS_COUNT];

    initArray(list, ARRAY_SIZE);

    for (int i = 0; i < THREADS_COUNT - 1; i++)
        pthread_create(&tid[i], NULL, threadSum, (void *)&i);

    int key = rand() % 200;
    pthread_create(&tid[2], NULL, threadSearch, (void *)&key);

    for (int i = 0; i < THREADS_COUNT - 1; i++)
        pthread_join(tid[i], NULL);

    int *isFound;
    pthread_join(tid[2], (void *)&isFound);

    printf("The Sum is: %d.\n", sum);
    (*isFound == -1)
        ? printf("The key: %d is not found\n", key)
        : printf("The key: %d is found at: %d\n", key, *isFound);

    return 0;
}
```

Output:

```
aimalexe@AimalKhans-PC:/mnt/d/computer_system_course/semester_5_(fall-23)/systems_programming/assignments
$ gcc question10.c -lpthread -o question10.o && ./question10.o
The Sum is: 2258.
The key: 11 is found at: 49
```

Question 11: *Write a multithreaded program for parallel file copying. Open both source files in master thread before creating threads*.

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
#include <string.h>
#include <pthread.h>
#include <sys/stat.h>

#define MAX_CHAR 256

struct threadCopyFileArgs
{
    int srcFile;
    int destFile;
};

void *threadCopyFile(void *arg)
{
    struct threadCopyFileArgs myArgs = *((struct threadCopyFileArgs
*)arg);

    char buff[MAX_CHAR];
    int readBites, writeBites;

    while (1)
    {
        if ((readBites = read(myArgs.srcFile, buff, MAX_CHAR)) == -
1)
        {
            fprintf(stderr, "Something went wrong while reading from
the src file: %d due to %s\n", myArgs.srcFile, strerror(errno));
            return NULL;
        }

        if ((writeBites = write(myArgs.destFile, buff, readBites))
== -1)
        {
            fprintf(stderr, "Something went wrong while writing to
the dist file: %d due to %s\n", myArgs.destFile, strerror(errno));
            return NULL;
        }

        if (readBites == 0)
            break;
    }

    printf("Copy completed from %d to %d\n", myArgs.srcFile,
myArgs.destFile);
```

```
}

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

    if ((argc % 2 == 0) || (argc == 1))
    {
        fprintf(stderr, "Usage:\n%s SRC_FILE DST_FILE [SRC_FILE
DST_FILE...]", argv[0]);
        return 1;
    }

    int fds[argc - 1];

    // open the files:
    for (int i = 1; i < argc; i++)
    {
        if (i % 2 != 0)
        { // 1. open sourc file (for reading only)
            if ((fds[i - 1] = open(argv[i], O_RDONLY)) == -1)
            {
                fprintf(stderr, "Something went wrong while opening
the source file: %s due to %s\n", argv[1], strerror(errno));
                return 1;
            }
        }
        else
        { // 2. open destination file if not present create it (for
writing only)
            if ((fds[i - 1] = open(argv[i], O_WRONLY | O_CREAT |
O_APPEND, S_IRWXU | S_IRWXG | S_IRWXO)) == -1)
            {
                fprintf(stderr, "Something went wrong while opening
the destination file: %s due to %s\n", argv[2], strerror(errno));
                return 1;
            }
        }
    }

    // create threads for each pair...
    int threadsCount = (argc - 1) / 2;
    pthread_t tid[threadsCount];
    struct threadCopyFileArgs thArgs;

    for (int i = 0; i < threadsCount; i++)
    {
        thArgs.srcFile = fds[2 * i];
        thArgs.destFile = fds[2 * i + 1];

        pthread_create(&tid[i], NULL, threadCopyFile, (void
*)&thArgs);
    }

    for (int i = 0; i < threadsCount; i++)
```

```
    {
        pthread_join(tid[i], NULL);
    }

    // close all files:
    for (int i = 0; i < argc - 1; i++)
        if (close(fds[i]) == -1)
        {
            perror("error while closing the file.");
            return 1;
        }

    return 0;
}
```

Output:

```
se/semester_5_(fall-23)/systems_programming/assignments$ gcc question11.c -lpthread -o question11.o && ./
question11.o f0.txt f1.txt f3.txt f4.txt
Copy completed from 5 to 6
Copy completed from 5 to 6
```

The End.