UCS1411 Operating Systems Lab
AY: 2021-22

Name: Krithika Swaminathan
Roll No.: 205001057

Assignment 2 – Simulation of System Commands using System Calls

Date: 14/03/2022

Program 1

Aim:

To develop a C program to simulate the cp command with the -i option using system calls.

Algorithm:

```
my_cp (-i):
    1. Start
    2. Check if an option (-i) is included in the command.
    3. If the difference between the number of arguments and the number of options is
       less than three, then print that two arguments are expected. If it is more than three,
       communicate to the user that there are too many arguments. If it is exactly three,
       then proceed.
    4. If an option is included, then set the 2nd argument as the source file and the 3rd
       argument as the destination file.
    5. If an option is not included, then set the 1st argument as the source file and the 2nd
       argument as the destination file.
    6. Open the source file and locate its file descriptor. If it already exists, proceed. If not,
       then report that the source file was not found.
    7. Copy the contents of the source file to a string.
    8. Open the destination file and do one of the following:
        1. If it does not exist, then create a new file and write the contents of the string
           to it.
        2. If it already exists and there is no (-i) option, then overwrite the contents of
           the string to the file.
        3. If it already exists and there is a (-i) option, then prompt the user to confirm
           before overwriting the file.
            • If the user says "yes", then overwrite the file.
            • If the user says "no" (or does not say yes), then exit the program
              without making any changes.
    9. Stop
```

Code:

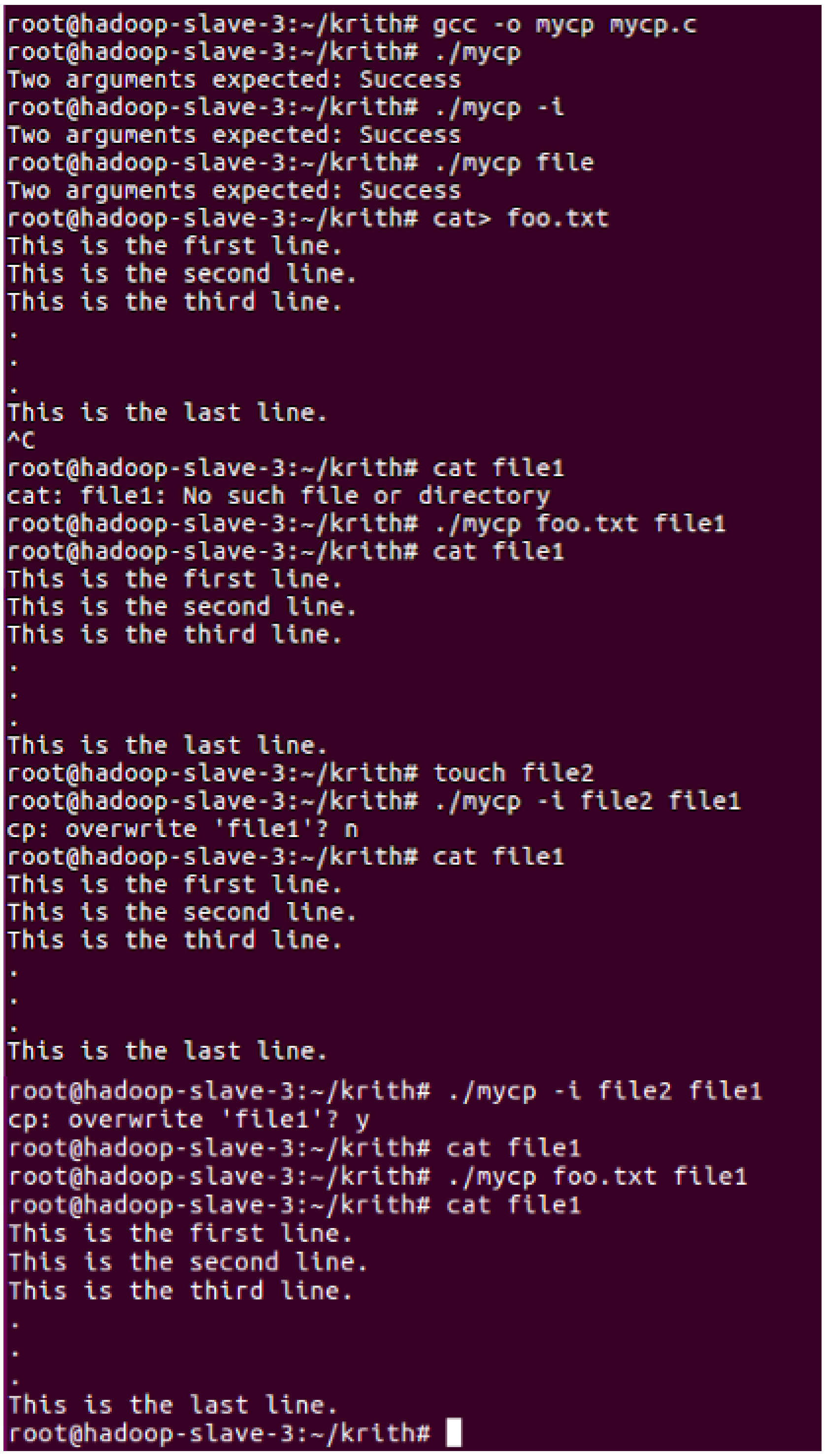
/* C program to simulate the cp [-i] command */

Name: Krithika Swaminathan
Roll No.: 205001057

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

#define MAX_LENGTH 100
#define MAX_BYTES 500

int main (int argc, char* argv[]){
        if (argc == 1) {
                perror("Two arguments expected"); exit(1);
                }

        int opCount = 0;
        if (strcmp(argv[1],"-i") == 0) {
                        opCount++;
                        }

        if (argc - opCount == 3){
                char src[MAX_LENGTH], dest[MAX_LENGTH], option[2];
                int src_fd, src_sz, dest_fd, dest_sz;
                char *c = (char *) calloc(100, sizeof(char));

                //check if command is to operating in interactive mode
                if (strcmp(argv[1],"-i") == 0) {
                        strcpy(src,argv[2]);
                        strcpy(dest,argv[3]);

                        //check if dest file exists
                        dest_fd = open(dest,O_RDONLY);
                        //ask the user if dest file is to be overwritten or not
                        if (dest_fd >= 0) {
                                char ch;
                                printf("cp: overwrite '%s'? ",dest);
                                scanf("%c",&ch);
                                if (ch != 'y' && ch != 'Y')
                                        exit(1);
                                }
                        }
                else {
                        strcpy(src,argv[1]);
                        strcpy(dest,argv[2]);
                        }

                //find file desciptor for src file
                src_fd = open(src,O_RDONLY);
                //include error msg for case when src file not found
                if (src_fd < 0) { perror("Command unsuccessful. Source file not found."); exit(1); }

                //save the contents of src file in a string
                src_sz = read(src_fd,c,MAX_BYTES);
                c[src_sz] = '\0';

                //close src file
                close(src_fd);

                //create dest file
                dest_fd = open(dest,O_WRONLY | O_CREAT | O_TRUNC,0754);
                //include error msg for case when dest file not created
                if (dest_fd < 0) { perror("Command unsuccessful. Destination file not created.");
exit(1); }

                //write the saved contents of the src file from the string to the dest file
                dest_sz = write(dest_fd,c,strlen(c));

                //close dest file
                close(dest_fd);
                }

        else if (argc - opCount > 3){
                perror("Too many arguments supplied"); exit(1);
                }

        else {
                perror("Two arguments expected"); exit(1);
                }
        }
```

Name: Krithika Swaminathan
Roll No.: 205001057

Output:



```
root@hadoop-slave-3:~/krith# gcc -o mycp mycp.c
root@hadoop-slave-3:~/krith# ./mycp
Two arguments expected: Success
root@hadoop-slave-3:~/krith# ./mycp -i
Two arguments expected: Success
root@hadoop-slave-3:~/krith# ./mycp file
Two arguments expected: Success
root@hadoop-slave-3:~/krith# cat> foo.txt
This is the first line.
This is the second line.
This is the third line.
.
.
.
This is the last line.
^C
root@hadoop-slave-3:~/krith# cat file1
cat: file1: No such file or directory
root@hadoop-slave-3:~/krith# ./mycp foo.txt file1
root@hadoop-slave-3:~/krith# cat file1
This is the first line.
This is the second line.
This is the third line.
.
.
.
This is the last line.
root@hadoop-slave-3:~/krith# touch file2
root@hadoop-slave-3:~/krith# ./mycp -i file2 file1
cp: overwrite 'file1'? n
root@hadoop-slave-3:~/krith# cat file1
This is the first line.
This is the second line.
This is the third line.
.
.
.
This is the last line.
root@hadoop-slave-3:~/krith# ./mycp -i file2 file1
cp: overwrite 'file1'? y
root@hadoop-slave-3:~/krith# cat file1
root@hadoop-slave-3:~/krith# ./mycp foo.txt file1
root@hadoop-slave-3:~/krith# cat file1
This is the first line.
This is the second line.
This is the third line.
.
.
.
This is the last line.
root@hadoop-slave-3:~/krith# 
```

UCS1411 Operating Systems Lab
AY: 2021-22

Name: Krithika Swaminathan
Roll No.: 205001057

Program 2

Aim:

To develop a C program to simulate the ls command with the -l option using system calls.

Algorithm:

```
my_ls (-l):
    1. Start
    2. Check if an option (-l) is included in the command.
    3. Find the difference between the number of arguments and the number of options.
        1. If it is 0, set the current directory to be the directory under consideration.
        2. If it is 1 or more, set each of the directories entered as an argument on the
           command line to be the set of directories under consideration.
    4. For each directory under consideration, get the entries in the directory structure.
    5. If option (-l) is included, then list the directory entries along with their file properties
       in long format. If not, then simply list the directory entries.
    6. Stop
```

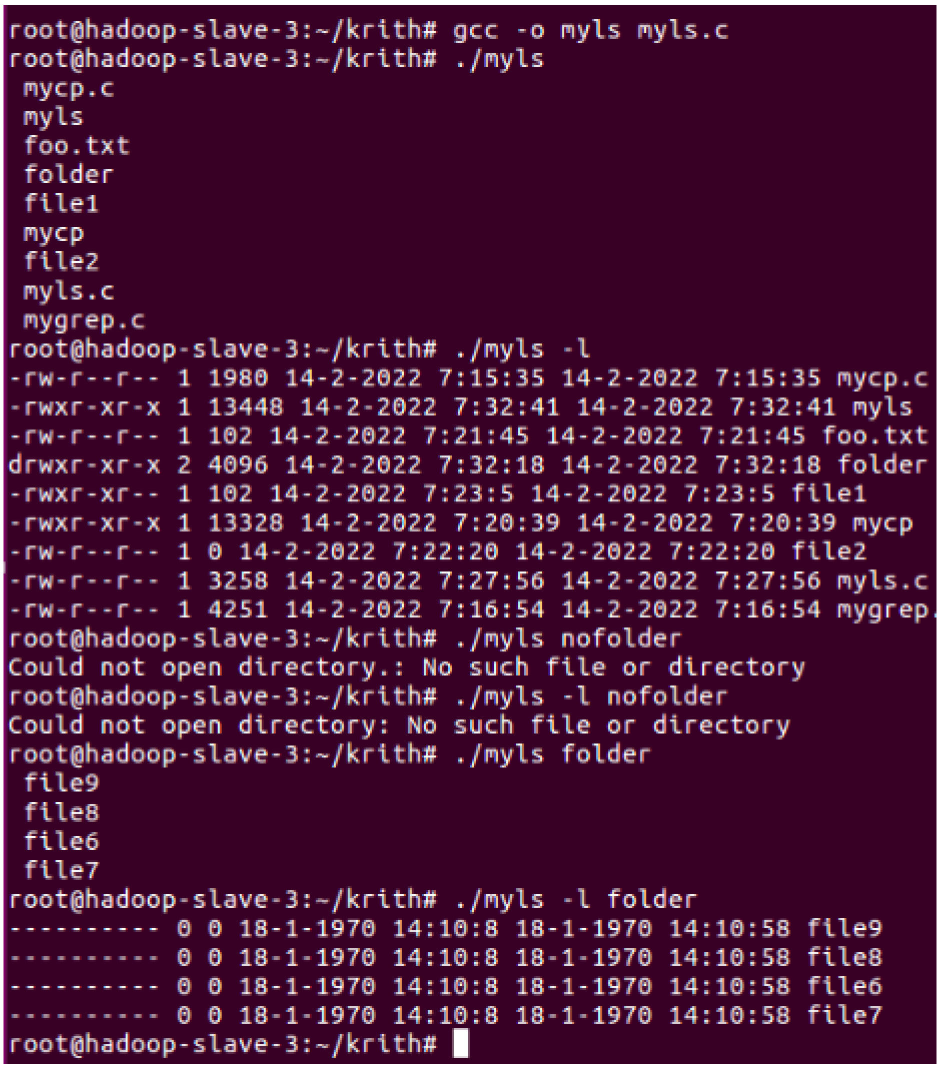
Code:

```
/* C program to simulate the ls [-l] command */
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <dirent.h>
#include <fcntl.h>
#include <string.h>
#include <time.h>

#define MAX_LENGTH 100
#define MAX_BYTES 500

void ls_dir (char*);
void ls_long (char*);
void ls_file (char*);
void printFileProperties(struct stat);

int main (int argc, char* argv[]) {
	if (argc == 1) {
		ls_dir(".");
		exit(1);
		}

	int opCount = 0;
	if (strcmp(argv[1],"-l")==0) {
		opCount = 1;
		if (argc-opCount == 1) {
			ls_long(".");
			}

		else if (argc-opCount == 2) {
			ls_long(argv[1+opCount]);
			}

		else {
			for (int i=1+opCount; i<argc; i++) {
				if (i==1+opCount)
					printf("%s: \n",argv[i]);
				else
					printf("\n%s: \n",argv[i]);
				ls_long(argv[i]);
				}
			}
		}

	else {
		if (argc-opCount == 1) {
			//call function to list contents of curr dir
			ls_dir(".");
			}

		else if (argc-opCount == 2) {
			ls_dir(argv[1]);
			}

		else {
			for (int i=1; i<argc; i++) {
				printf("%s: \n",argv[i]);
				ls_dir(argv[i]);
				}
			}
		}
	}

//function to list contents of a given dir
void ls_dir (char* dir) {
	DIR *folder;
	struct dirent *entry;
	//open given dir
	folder = opendir(dir);
	if (folder == NULL) {
		perror("Could not open directory.");
		exit(1);
		}
	//read and print contents of dir
	while (entry = readdir(folder)) {
		if (strcmp(entry->d_name,".")!=0 && strcmp(entry->d_name,"..")!=0)
			printf(" %s\n", entry->d_name);
		}

	//close dir
	closedir(folder);
	//printf("\n");
	}

//function to list contents of a given dir in long format
void ls_long (char* dir) {
	DIR *folder;
	struct dirent *entry;
	struct stat fileStats;

	//open given dir
	folder = opendir(dir);
	if (folder == NULL) {
		perror("Could not open directory");
		exit(1);
		}
	//read and print contents of dir
	while (entry = readdir(folder)) {
		if (strcmp(entry->d_name,".")!=0 && strcmp(entry->d_name,"..")!=0) {
			//ls_file(entry->d_name);
			stat(entry->d_name,&fileStats);
			printFileProperties(fileStats);
			printf("%s\n",entry->d_name);
			}
		}

	//close dir
	closedir(folder);
	}

//function to print the file properties
void printFileProperties(struct stat stats) {
	struct tm dt;

	//printf("%ld ", stats.st_ino);
```

```
        //printf("%o ", stats.st_mode);

        printf( (S_ISDIR(stats.st_mode)) ? "d" : "-");
        printf( (stats.st_mode & S_IRUSR) ? "r" : "-");
        printf( (stats.st_mode & S_IWUSR) ? "w" : "-");
        printf( (stats.st_mode & S_IXUSR) ? "x" : "-");
        printf( (stats.st_mode & S_IRGRP) ? "r" : "-");
        printf( (stats.st_mode & S_IWGRP) ? "w" : "-");
        printf( (stats.st_mode & S_IXGRP) ? "x" : "-");
        printf( (stats.st_mode & S_IROTH) ? "r" : "-");
        printf( (stats.st_mode & S_IWOTH) ? "w" : "-");
        printf( (stats.st_mode & S_IXOTH) ? "x" : "-");
        printf(" ");

        printf("%ld ", stats.st_nlink);

        printf("%ld ", stats.st_size);

        //get file creation time in seconds and convert seconds to date and time format
        dt = *(gmtime(&stats.st_ctime));
        printf("%d-%d-%d %d:%d:%d ", dt.tm_mday, dt.tm_mon, dt.tm_year + 1900, dt.tm_hour,
dt.tm_min, dt.tm_sec);

        // File modification time
        dt = *(gmtime(&stats.st_mtime));
        printf("%d-%d-%d %d:%d:%d ", dt.tm_mday, dt.tm_mon, dt.tm_year + 1900, dt.tm_hour,
dt.tm_min, dt.tm_sec);
        }
```

Name: Krithika Swaminathan
Roll No.: 205001057

Output:



```
root@hadoop-slave-3:~/krith# gcc -o myls myls.c
root@hadoop-slave-3:~/krith# ./myls
 mycp.c
 myls
 foo.txt
 folder
 file1
 mycp
 file2
 myls.c
 mygrep.c
root@hadoop-slave-3:~/krith# ./myls -l
-rw-r--r-- 1 1980 14-2-2022 7:15:35 14-2-2022 7:15:35 mycp.c
-rwxr-xr-x 1 13448 14-2-2022 7:32:41 14-2-2022 7:32:41 myls
-rw-r--r-- 1 102 14-2-2022 7:21:45 14-2-2022 7:21:45 foo.txt
drwxr-xr-x 2 4096 14-2-2022 7:32:18 14-2-2022 7:32:18 folder
-rwxr-xr-- 1 102 14-2-2022 7:23:5 14-2-2022 7:23:5 file1
-rwxr-xr-x 1 13328 14-2-2022 7:20:39 14-2-2022 7:20:39 mycp
-rw-r--r-- 1 0 14-2-2022 7:22:20 14-2-2022 7:22:20 file2
-rw-r--r-- 1 3258 14-2-2022 7:27:56 14-2-2022 7:27:56 myls.c
-rw-r--r-- 1 4251 14-2-2022 7:16:54 14-2-2022 7:16:54 mygrep.
root@hadoop-slave-3:~/krith# ./myls nofolder
Could not open directory.: No such file or directory
root@hadoop-slave-3:~/krith# ./myls -l nofolder
Could not open directory: No such file or directory
root@hadoop-slave-3:~/krith# ./myls folder
 file9
 file8
 file6
 file7
root@hadoop-slave-3:~/krith# ./myls -l folder
---------- 0 0 18-1-1970 14:10:8 18-1-1970 14:10:58 file9
---------- 0 0 18-1-1970 14:10:8 18-1-1970 14:10:58 file8
---------- 0 0 18-1-1970 14:10:8 18-1-1970 14:10:58 file6
---------- 0 0 18-1-1970 14:10:8 18-1-1970 14:10:58 file7
root@hadoop-slave-3:~/krith#
```

Name: Krithika Swaminathan
Roll No.: 205001057

Program 3

Aim:

To develop a C program to simulate the grep command with the -c, -n and -v options using system calls.

Algorithm:

```
my_grep (-c,-n,-v):
    1. Start
    2. Check if there is at least one argument. If not, exit.
    3. Check for options that are included in the command and validate them.
    4. If the difference between the number of arguments and the number of options is
       less than three, report that the number of arguments is less. If not, proceed.
    5. Set the argument immediately succeeding the command name (and options, if any)
       to be the pattern that is to be searched for.
    6. Set the rest of the arguments to be the files that are to be searched.
    7. Locate the file descriptor of each file. If a file is not found, report an error.
    8. Read the contents of each file in a string.
    9. Search the string for substrings that match the given pattern.
    10. Do one of the following:
        1. If no option is present, display the lines containing matches of the pattern.
        2. If option (-c) is given, display the frequency of pattern matches in the string.
        3. If option (-n) is given, display the lines containing matches of the pattern
           along with their line numbers.
        4. If option (-v) is given, display the lines which do not contain matches of the
           pattern.
        5. If a combination of these options is given, display the corresponding
           combined output as required.
    11. Stop
```

Code:

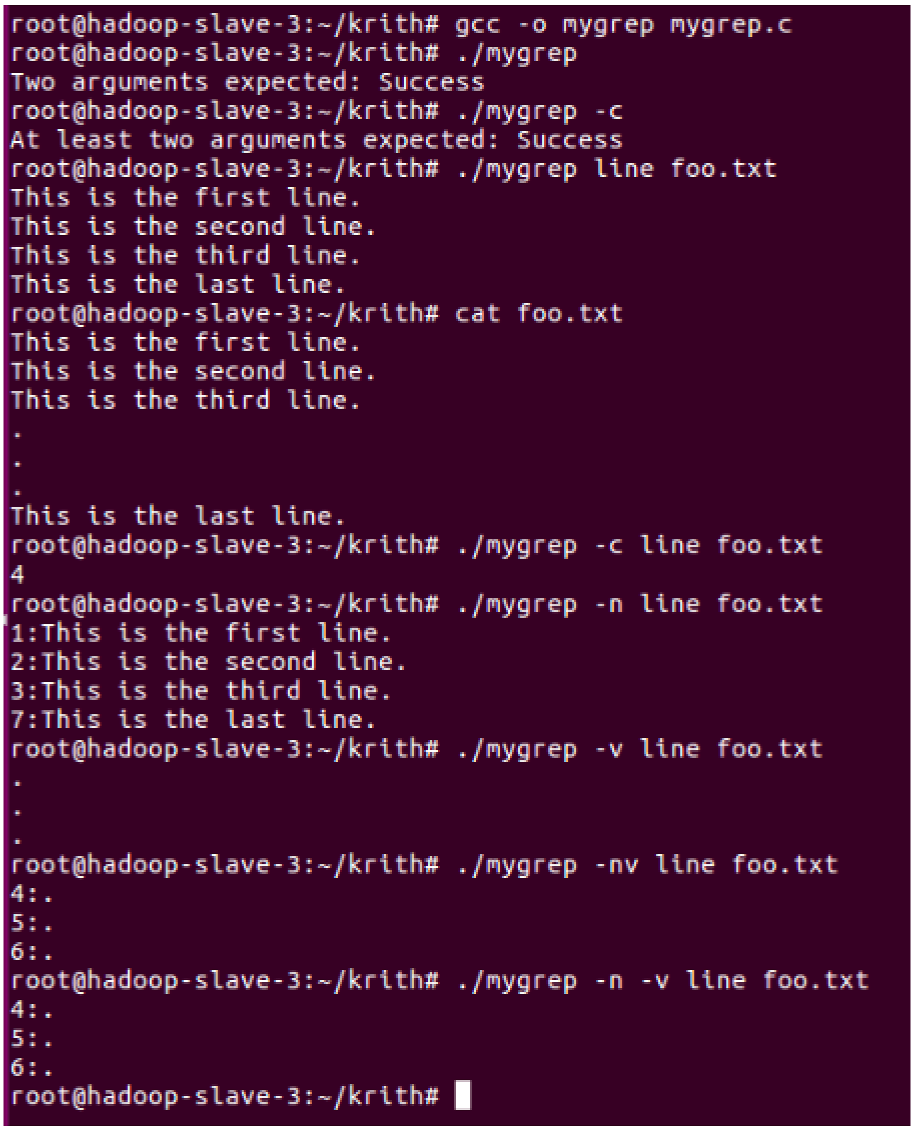
```c
/* C program to simulate the grep [-cvn] command */
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <dirent.h>
```

Name: Krithika Swaminathan
Roll No.: 205001057

```
#include <fcntl.h>
#include <string.h>

#define MAX_LENGTH 100
#define MAX_BYTES 10000
#define MAX_OPTIONS 3

int substring_count (char*, char*);

int main (int argc, char* argv[]){
	if (argc == 1) {
		perror("Two arguments expected"); exit(1);
		}

	int opCount = 0;
	for (int i=1; i<argc && (argv[i][0]=='-' && argv[i][1]!='\0'); i++) {
		opCount++;
		}

	if (argc - opCount >= 3){
		int src_fd, src_sz, num = MAX_OPTIONS;
		char src[MAX_LENGTH], pattern[MAX_LENGTH], options[num];
		char *content = (char *) calloc(MAX_BYTES, sizeof(char));
		memset(options,'-',num*sizeof(options[0]));

		//determine which options have been included
		int k=0;
		for (int i=1; i<argc && (argv[i][0]=='-' && argv[i][1]!='\0'); i++) {
			for (int j=1; argv[i][j]!='\0'; j++) {
				switch (argv[i][j]) {
					case 'n': {
						options[0] = argv[i][j];
						k++; break;
						}
					case 'v': {
						options[1] = argv[i][j];
						k++; break;
						}
					case 'c': {
						options[2] = argv[i][j];
						k++; break;
						}
					default: {
						perror("Invalid option. Exiting..."); exit(1);
						break;
						}
					}
				}
			}
```

```
    //identify the pattern from CLA
    strcpy(pattern,argv[1+opCount]);

    for (int i = opCount+2; i<argc; i++) {
        //identify the filename from CLA
        strcpy(src,argv[i]);

        //find file desciptor for src file
        src_fd = open(src,O_RDONLY);
        //include error msg for case when src file not found
        if (src_fd < 0) { perror("Command unsuccessful"); exit(1); }

        //save the contents of src file in a string
        src_sz = read(src_fd,content,MAX_BYTES);
        content[src_sz] = '\0';

        //close src file
        close(src_fd);

        //display required output
        if (k==0) {
            //split the content into lines
            char* line = strtok(content,"\n");
            while (line != NULL) {
                //check if pattern is a substring
                    //note: match regex too
                    //regcomp and regex functions

                if (strstr(line,pattern))
                    printf("%s\n", line);
                //go to the next line
                line = strtok(NULL,"\n");
                }
            }

        else if (options[2] == 'c') {
            int count = 0;
            //split the content into lines
            char* line = strtok(content,"\n");
            while (line != NULL) {
                //check if pattern is a substring
                if (strstr(line,pattern)) {
                    count += substring_count(line,pattern);
                    }
                //go to the next line
                line = strtok(NULL,"\n");
                }
            //print the number of occurrences
				printf("%d\n",count);
				}

		else if (options[0] == 'n' && options[1] == '-') {
				int n = 0;
				//split the content into tokens on newline
				char* line = strtok(content,"\n");
				while (line != NULL) {
					//count the line numbers
					n++;
					//check if pattern is a substring
					if (strstr(line,pattern))
						printf("%d:%s\n", n,line);
					//go to the next line
					line = strtok(NULL,"\n");
					}
				}

		else if (options[0] == 'n' && options[1] == 'v') {
				int n = 0;
				//split the content into tokens on newline
				char* line = strtok(content,"\n");
				while (line != NULL) {
					//count the line numbers
					n++;
					//check if pattern is a substring
					if (!strstr(line,pattern))
						printf("%d:%s\n", n,line);
					//go to the next line
					line = strtok(NULL,"\n");
					}
				}

		else if (options[0] == '-' && options[1] == 'v') {
				//split the content into tokens on newline
				char* line = strtok(content,"\n");
				while (line != NULL) {
					//check if pattern is a substring
					if (!strstr(line,pattern))
						printf("%s\n",line);
					//go to the next line
					line = strtok(NULL,"\n");
					}
				}

		else { perror("Search unsuccesful"); exit(1); }

		}
	}
	else {
		perror("At least two arguments expected"); exit(1);
		}
	}

//function to count the frequency of a given substring in a string
int substring_count(char* string, char* substring) {
	int i, j, l1, l2;
	int count = 0;
	int found = 0;

	l1 = strlen(string);
	l2 = strlen(substring);

	for(i = 0; i < l1-l2 + 1; i++) {
		found = 1;
		for(j = 0; j < l2; j++) {
			if(string[i+j] != substring[j]) {
				found = 0;
				break;
				}
			}
		if(found) {
			count++;
			i = i + l2 -1;
			}
		}
	return count;
	}
```

Output:



```
root@hadoop-slave-3:~/krith# gcc -o mygrep mygrep.c
root@hadoop-slave-3:~/krith# ./mygrep
Two arguments expected: Success
root@hadoop-slave-3:~/krith# ./mygrep -c
At least two arguments expected: Success
root@hadoop-slave-3:~/krith# ./mygrep line foo.txt
This is the first line.
This is the second line.
This is the third line.
This is the last line.
root@hadoop-slave-3:~/krith# cat foo.txt
This is the first line.
This is the second line.
This is the third line.
.
.
.
This is the last line.
root@hadoop-slave-3:~/krith# ./mygrep -c line foo.txt
4
root@hadoop-slave-3:~/krith# ./mygrep -n line foo.txt
1:This is the first line.
2:This is the second line.
3:This is the third line.
7:This is the last line.
root@hadoop-slave-3:~/krith# ./mygrep -v line foo.txt
.
.
.
root@hadoop-slave-3:~/krith# ./mygrep -nv line foo.txt
4:.
5:.
6:.
root@hadoop-slave-3:~/krith# ./mygrep -n -v line foo.txt
4:.
5:.
6:.
root@hadoop-slave-3:~/krith# 
```

Learning outcomes:

- Linux commands were replicated by programming in C.
- The inner workings of the given commands were understood while implementing them.
- Validation of input and error handling was implemented.