CSSE2310: 2013 exam answers

UQAttic

This document has been has yet to have been completed

Question 1) Write shell commands to do the following:

[10 marks (2 each)]

A) Delete all files with names beginning with A and ending in .c.

rm –rf A*.c

B) Show all lines in the file stuff which start with W:

Cat stuff | grep “^W”

C) A file nums consists of 4 space separated columns.
Output columns 1, 3, 4 sorted by the last column.

cut -d ‘ ‘ -f1,3,4 nums | sort -k 3

OR

sort -k 4 nums | cut -d’ ‘ -f1,3,4

OR

cat nums | cut -d’ ‘ -f1,3,4 | sort -k 3

D) Create a file c.c which is a copy of b.c.

cp b.c c.c

E) For files f 1,f 2,f 3 show all lines from any of them which contain all the words “song”, “river” and
“terrible”.

Grep song f1 f2 f3 | Grep song f1 f2 f3 | etc.

or

grep “song\|river\|terrible” f1 f2 f3

Question 2) Write C to declare foo as . . . : [5 marks (1 each)]

A) an array of 12 integers.

Int foo[12];

B) a pointer to a positive integer.

Int* foo = 3;

C) another name for a small integer.

Short foo;
^ I think they are asking for a typedef, not a simple short int declaration

typedef short int foo;

D) a struct containing an integer called i and a string called s.

```
Struct foo {

int I;
char* s;

};
```

E) a pointer to a function which takes two floating point values and returns a string.

```
char * (*foo) (float I, float j) {

…..
}
```

Question 3) What is the output from the following statements [11 marks (1 each)]

A)

```
int a=3; int b=7 printf("%d", b/a);
```

2

B)

```
int a=3;
int b=7;
printf("%d %d", b^a, b|2);
```

4 7

C)

```
int a=3;
int b=7;
printf("%d %d", a++, --b);
```

3 6

D)

```
int a=3;
int b=7;
printf("%d", a+b*2);
```

17

E)

```
int a=3;
int b=7;
int c=(a>b)? a : b+2;
printf("%d", c);
```

9

F
)

```
int a=3; int b=7;
do {
a=a+1;
b-=1;
} while(a>b);
printf("%d %d", a, b);
```

4 6

G)

```
int a=3;
int b=7;
for (int i=0; i<2; ++i )
        for (int j=i;j<4; ++j ) {
            if (j>2) {
              break;
            }
          a++; b--;
}

printf("%d %d", a, b);
```

5 6

Would not compile as there is no semi colon on the third line?

^^ Answer is not even possible, since a & b are in/decremented together. Correct answer is 8 2. [BM]

H)

```
int a=3;
int b=7;
int c=4*(3,7);
printf("%d", c);
```

28 ?

I can confirm 28.

I)

```
int   a=3;
int b=7; int c=12 if
(b>c) c--;
b++;
if (c & b)
c-=3;
printf("%d %d", b, c);
```

missing ; after int c declaration but 8 9

J)

```
int a[]={3,4,5};
int b=7;
int* x=a;
int c=(*(++x))--;
printf("%d %d %d %d", c, a[0], a[1], a[2]);
```

4 3 3 5

K)

```
char a[]="world";
char* b=a+3;
a[1]=0;
printf("%s %s", a, b);
```

w ld

Question 4) A system has 32bit virtual addresses, 4KB pages and page table entries are 4Bytes. It uses a two level page table.
[6 marks (2 each)]

A) Which pages do the following (decimal) addresses belong to?
11111, 22222, 9001, 404040

2 , 5 , 2 , 98

Consider, 4096 bytes in a page
Page = virtual address / page size
Page = 11111 / 4096 = 2.7
Page = 2

This means that the virtual address of 11111 is on the THIRD page, because the numbering starts from 0.

B) What causes page faults?

When an object/process/program is on disk but not in memory.

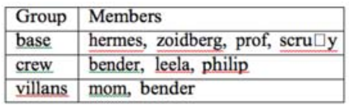
C) What causes segmentation faults?

access invalid page
access read only page

Question 5)
[11 marks] Consider the following directory listing:

```
total 808
2244723 drwxr-xrwx    6 hermes base      4096 Sep  9 11:13 .
2228225 drwxrwxrwx  409 root   root    319488 Sep  9 11:13 ..
2228804 -rw-r--r--    1 hermes base        66 Sep  9 10:32 Makefile
2228798 -rw-r--r--    1 hermes crew     83737 Sep  9 10:32
ass1_spec.pdf
2228802 -rw-rw-r--    1 hermes crew     17485 Sep  9 10:32
ass1_spec.tex
2228908 -rw-r--r--    1 hermes crew     54245 Sep  9 10:34
ass2spec.pdf
2228911 -r--rw-rw-    1 hermes base      3524 Sep  9 10:34
ass2spec.tex
2228914 -r--rw-rw-    2 hermes villans  18615 Sep  9 10:41
ass3_spec.tex
2253442 drwxr-xr--    2 hermes base      4096 Sep  9 10:46 fireflies
2228914 -rw-r--r--    2 hermes base     18615 Sep  9 10:41 herring
2228920 -rw-r--r--    1 hermes crew       340 Sep  9 10:32 marks
2228949 -rwxr-xr-x    1 hermes villans    749 Sep  9 10:32 mkres
2253440 drwx---r-x    2 hermes villans   4096 Sep  9 10:34 procmarks
2229358 -rwxr--r--    1 hermes crew    224787 Sep  9 10:43 program
2228904 -rw-r-----    1 hermes base       932 Sep  9 10:33 stuff.txt
2228950 -rwxr-xr-x    1 hermes villans  19592 Sep  9 10:32 thebox
```

```
2228988 -rw-r--r--    1 hermes uusers    9343 Sep  9 10:32 thebox.c
2228907 lrwxrwxrwx    1 hermes crew        5 Sep  9 10:38 things ->
zorro
2253446 drwxr-xr-x    3 hermes villans   4096 Sep  9 11:11 toronado
2253441 drwxr-xr-x    3 hermes crew      4096 Sep  9 11:08 zorro
```



Group	Members
base	hermes, zoidberg, prof, scru□y
crew	bender, leela, philip
villans	mom, bender

A) What can zoidberg do to the following:
[2 marks]

- stuff.txt

  read

- procmarks

  Read and execute

B) Which users can modify all of the .tex files (without changing the permissions)?
[2 marks]

The Crew usergroup

C) What command(s) could mom type to execute program?
[2 marks]

She can't as only Hermes has program execute permissions
She could copy the file program "cp program program2", making her the owner of program2 and then "chmod +x program2" and execute program2, this isn't strictly executing "program" though, only a copy of it.

D) What would change in the directory listing after hermes executed rmdir toronado (and why?)
[2 marks]

-If it was an empty directory, the directory would be deleted and the inode count will decrease.

-But, if the directory has contents within it, then it would not be deleted.

It can be seen from the reference count, which is 3, that the directory is not empty, so the directory will not be deleted and therefore nothing will be changed. Note that empty directories have a reference count of 2. [BM]

E) What command was used to create things?
[1 marks]

Ln –s zorro things

F) Given the following commands and their output:

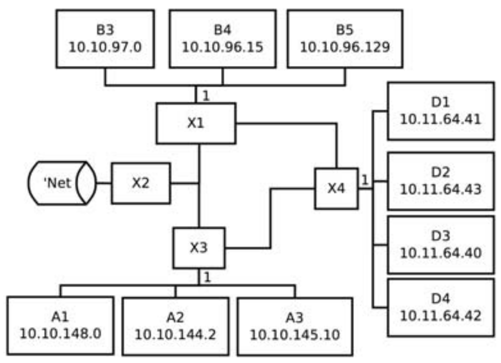
```
prompt> ls -l zorro/transport
lrwxrwxrwx 1 hermes base 21 Sep  9 11:20 zorro/transport ->
../fireflies/serenity

prompt> ls -l fireflies/serenity
-rwx--x--x 1 hermes base 1072966 Sep  9 10:46 fireflies/serenity
```

Can bender run ./things/transport ? (Why?)

No because bender is in a different user group, and does not have access rights.

Question 6)
[13 marks] Consider the following network. If addresses are required but not given, you may choose an address
that makes the subnet as small as possible [in terms of number of possible hosts].



A) What are the network addresses, subnet masks and broadcast addresses for the subnets attached to the following interfaces? Use the smallest possible subnets.
[5 marks]

	Network Address	Subnet Mask	Broadcast Address
X1 (interface 1)	10.10.96.0	255.255.254.0	10.10.97.255
X3 (interface 1)	10.10.144.0	255.255.248.0	10.10.151.255
X4 (interface 1)	10.11.64.32	255.255.255.240	10.11.64.47

Brown = Hayden.
For X1:

$$\delta = \max\{BX\} - \min\{BX\} = B_3 - B_4$$

So comparing these in binary we have:

$$97 = 64 + 32 + 1 = 01100001_2$$
$$96 = 64 + 32 = 01100000_2$$

These are identical for the first 7 bits (starting at the MSB). Hence the length of the subnet mask will be $l = 8 + 7 = 15$. Therefore the subnet mask S_m is given by the following:

$$S_m = 255.255.254.0$$

Now the network address can be found by bitwise ANDing the subnet mask with any element from BX.

$$N_a = AND(S_m, BX)$$ (pick the easiest for BX)
$$N_a = AND(255.255.254.0,\ 10.10.97.0) = 10.10.96.0$$

Finally the broadcast address, B_a can be determined by bitwise ORing the network address and the
bitwise inverse of the subnet mask. Hence:

$$B_a = OR(N_a, S_m{}^{-1})$$
$$B_a = OR(10.10.96.0,\ 0.0.1.255)$$
$$B_a = 10.10.97.255$$

For X4:

$$\delta = \max\{DX\} - \min\{DX\} = D_2 - D_3$$

Once again, comparing these in binary yields the following:

$$43 = 32 + 8 + 2 + 1 = 00101011_2$$
$$40 = 32 + 8 = 00101000_2$$

So this one differs at the 6th (I believe I differs at the 7th, which changes all the calculations below…) bit, so the length of the subnet mask this time will be $l = 8 + 6 = 14$. Hence the subnet mask is:

$$S_m = 255.255.255.252$$

Using the same approach above to finding the network address we come across a problem:

$$N_a = AND(S_m, DX)$$
$$N_a = AND(255.255.252,\ 10.11.64.40)$$
$$N_a = 10.11.64.40$$

Notice that this network address is not unique. So the subnet mask must be modified such that it is unique. Referring back to the binary representations of 43 and 40, find the first one (which will change the subnet mask) which now changes the length to $l = 8 + 4 = 12$. The new subnet mask will therefore be:

$$S_m = 255.255.255.240$$

The correct network address will be:

$$N_a = AND(S_m, DX) = 10.11.64.32$$

The broadcast address will be:

$$B_a = OR(N_a, S_m{}^{-1}) = 10.11.64.47$$

For X3, the same process applies you should get the following:

$$S_m = 255.255.248.0$$

N_a = 10.10.144.0
B_a = 10.10.151.255

B) What task does the bind() function perform?
[1 mark]

To put it simply, bind says to the system : okay, from now on, any packet with destination {address->sun_addr} should be forwarded to my socket_fd, so I can read them.

C) X2 perfoms NAT for this network. What is NAT and why is it necessary?
[2 marks]

answer 1: Network Address Translation

It is necessary because method of modifying network address information in Internet Protocol (IP) datagram packet headers while they are in transit across a traffic routing device for the purpose of remapping one IP address space into another.

For example assign a company a single IP address, then use unique private IP addresses within the company, then change the private IP address back to the companies IP address when data packet leaves the network.

answer2:
what: network address translation
why: all computer are given a private address on a network (10.x.x.x, 172.16.x.x,192.168.x.x). These Ip addresses cannot be used to access the internet. NAT performs a translation from private address to public address , so computers can access the internet.

D) To which layers do the following belong:
[3 marks]

Term	Layer
Mac Address	Data link layer
Socket	Session layer (SSL) / Application layer
IP Address	Network layer - for the internet protocol
Port	Transport layer
UDP	Transport layer
URL	Application layer

Possible layers:
metaphysical, application, web, physical, network, transport, gooey caramel, putty, link, wifi.

E) What is the purpose of a gateway address?

The gateway default address is a router that is the face of a local area network which sends packets out.
When you try to access www.google.com , your computer will try to see if that address is in the local network before talking with your router to send the packets out of the local network.

[2 marks]

Question 7) Consider a “unix” filesystem where: [8 marks (2 each)]

• i-nodes have 10-direct pointers, 1 indirect pointer and 1 double indirect pointer.

• Blocks are 8KB

• Block pointers are 4Bytes

• blocks are numbered from 0.

A) Why is fragmentation a problem for linked filesystems but not for indexed filesystems?

Indexed filesystems are sequential so adjacent blocks store data “in order”. Linked filesystems hold pointers to dats, so the data can be spread (physically) across the disk.

B) How many blocks (in total) must be accessed to read the following blocks from a file: 9, 2053, 2057

1 for 9 and 2 for 2053 and 1 for 2057 (the pointer is stored from 2053 read) so all together, a total of 4.

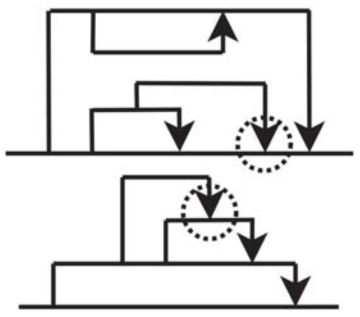
C) What is the maximum possible file size for this file system?

total size = (10*8192) + (1*2048*8192) + (1*2048^2 * 8192) = 33,570,896 kB

D) If an additional 2 double indirect pointers were added to the inodes on this filesystem, what would be the increase in maximum file size?

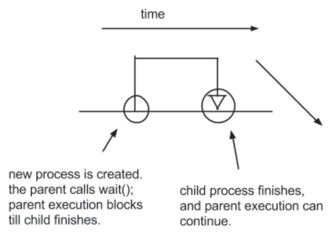
Increase is 2*2048^2 * 8192 = 67,108,864 kB

Question 8)
[10 marks]



A) Consider the following process fork diagrams:
The circled wait in the top diagram is possible. The circled wait in the lower diagram is not possible. Why?

if you can represent a process forking like this:


time
new process is created. the parent calls wait(); parent execution blocks till child finishes.
child process finishes, and parent execution can continue.

then, in the second diagram:
a new process is created.

But, the problem in the second diagram is that the child process returns to a process that hasn't been created yet.

In the second diagram it appears that it wasn't repead by it's parent, nor the init process. If we assume the downward arrow is the reaping(ending of the process). The first diagram appears to be repead by init (as it's parent has died)

[3 marks]

B) Which C function can be used to test if a child process has terminated?
[1 mark]

WAITPID

C) Which function is used to send a signal to a process?
[1 mark]

answer 1: Kill.
answer 2: exit

^^exit does not send a signal *to* a process, but rather exits the current process with the given exit code, so it is not a valid answer. [BM]

D) Which pthreads functions are used to perform the following tasks:
[2 marks]

1. terminate a thread

answer1: pthread_exit
answer 2: pthread_cancel

2. retrieve the exit status of a thread.

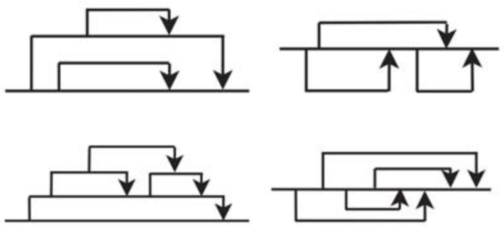
the exit status can be obtained by another thread by calling pthread_join.

E) Consider the following functions:
[2 marks]

```
void* f(void* v) {
    // X
    void* res=doThings(v);
    // Y
    return res;
}
```

Which functions could be called at X and Y to ensure that only one thread at a time executes
doThings()?

X: pthread_mutex_lock
Y: pthread_mutex_unlock
or
X: sem_wait
Y: sem_post



F) Which of the following pthread diagrams are possible? (Circle the possible ones)
[1 mark]

All expect bottom left. a thread is joined inside another thread.

Question 9) In all of the following you may omit #includes. You may assume that all system calls succeed and that all processes exit normally. You may assume that all lines in files have 79 or fewer characters. You may assume that none of the strings to be searched for contain special charcters for grep. That is, no escaping is required.
[26 marks]

A) Implement a function matchingLines which takes a string and a filename and prints (to stdout) all the lines in the file which contain the string. It should return true if the file could be opened successfully and false if not. Hint: make use of standard string functions.
[4 marks]

Answer 1:

bool matchingLines(const char* string, const char* filename) {

This is my solution. It compiles fine, I just haven't tested it for functionality. [Sam M.]

- changed ordering of strstr() args (they were backwards) [Matt S.]

```
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

bool matchingLines(const char* string, const char* filename) {
        FILE *fp;
        char buffer[100];
        memset(buffer, '\0', strlen(buffer)); //memset buffer to avoid leaks
        fp = fopen(filename, "r+");
```

```
        if (fp == NULL) {     //check for success of opening file
                return false;
        }
        while (fgets(buffer, 100, fp) != NULL){   //Keep reading lines while they exist
                if (strstr(buffer, string) != NULL) {     //Check the line for string
                        printf("%s", buffer);
                }
                memset(buffer, '\0', strlen(buffer)); //reset buffer for next read from file
        }
        fclose(fp);
        return true;
}
```

^^ No need to use memset to clear buffer, since it is just overwritten by fgets. [BM]

Answer 2:

```
#include <stdio.h>
#include <string.h>
#include <stdbool.h>
bool matchingLines(const char* string, const char* filename) {
                {
                        char temp[79];
                        FILE * f;
                        if ((f = fopen(filename, "r")) == NULL){
                                return false;
                        }

                        while (fgets(temp, 79, f) && (temp != NULL))
                        {
                                if (strstr(temp, string)){
                                        printf("%s", temp);
                                }

                        }
                        fclose(f);
                        return true;
                }

}
```

^^ A line could be 79 characters, so buffer needs to be able to hold 79 characters for line + 1 for terminating newline + 1 for string null terminator, so buffer needs to be able to hold at least 81 characters, not 79. [BM]

Answer 3: [BM]
bool matchingLines(const char* string, const char* filename) {

```
        FILE *fp = fopen(filename, “r”);

        if(!fp) {
                return false;
        }

        char buf[81];

        while(fgets(buf, 81, fp) != NULL) {
                if(strstr(buf, string) != NULL) {
                        printf(“%s”, buf);
                }
        }

        fclose(fp);
        return true;
}
```

B) Implement a function grepSearch which takes a string and a filename and prints (to stdout) all the lines in the file which contain the string. It will do this by invoking the grep program (which will be on the path). It should return true if the file could be opened successfully and false if not. You may not use the functions system(), or popen(). Hint: grep has an exit status of 2 if the file can not be opened. It will never have a status higher than 2. [4 marks]

```
  bool grepSearch(const char* string, const char* filename) {
```

- my solution, seems to work… [matt s.]
- Possible problem: This solution doesn’t redirect stderr from grep to /dev/null if a invalid file is given to grep it will output “grep: FILE: No such file or directory”
- good point - added.

```
bool grepSearch(const char* string, const char* filename) {
        int pid, childPid, status, pipeFds[2];

        pipe(pipeFds);
        pid = fork();

        if (pid != 0) { /* parent */
                close(STDOUT_FILENO); //close stdout
                dup(pipeFds[1]); //make stdout pipeFds[1]
                close(pipeFds[0]); //dont need

                childPid = wait(&status);
                if (WEXITSTATUS(status) == 2) {
                        return false;
                }
                return true;
        } else { /* child */
                close(STDIN_FILENO);
                dup(pipeFds[0]); // make stdin pipeFds[0]
                int devNull = open(“/dev/null”, O_WRONLY);
                dup2(devNull, STDERR_FILENO);
                close(pipeFds[1]); //dont need
                char *arg_list[] = {"grep", (char *)string, (char *)filename, NULL};
                execvp("grep", arg_list);
        }
}
```

^^ stdout is automatically redirected to stdout of the parent, so there is no need for redirection of file descriptors. The following solution works fine:

```
bool grepSearch(const char* string, const char* filename) {
    pid_t pid;

    if((pid = fork()) == 0) { // child
```

```
        execlp("grep", "grep", string, filename, NULL);
    }

    // parent
    int child_status;
    waitpid(pid, &child_status, 0);

    return !(WIFEXITED(child_status) && WEXITSTATUS(child_status) == 2);
}
```

Further, there is no real requirement to redirect stderr to the bitbucket because it is not stated as a specific requirement (i.e. it is undefined). [BM]

C) Implement a function matchingLinesMany which takes a string and an array of filenames (and its size) and prints (to stdout) all the lines in all the files which contain the string. All the specified files should be processed concurrently. If any of the files result in an error, then matchingLinesMany will return false otherwise return true. Hint: You should call the function from part A. For this part, you can assume it exists and works. [6 marks]

```
  bool  matchingLinesMany(const char* string, const char** filenames, int numfiles) {
```

Answer 1

```
void* do_thread(void *arg) {
        char** args = (char **)args;

        return (void *)matchingLines(args[0], args[1]);
}

bool matchingLinesMany(const char* string, const char** filenames, int numfiles) {
        pthread_t ids[numfiles];

        for (int i = 0; i < numfiles; i++) {
                char* args[2] = {(char *)string, (char *)filenames[i]};
                pthread_create(&ids[i], NULL, do_thread, (void *)args);
        }
        bool success = true;
        for (int i = 0; i < numfiles; i++) {
                bool retVal = true;
                pthread_join(ids[i], (void**)&retVal);
                if (retVal == false) {
                        success = false;
                }
        }
        return success;
}
```

^^ pthread_t ids[numfiles] is not valid C, since numfiles is a variable. pthread_t *ids = malloc(sizeof(pthread_t) * numfiles) is (with a corresponding free at the end). [BM]

Alternative answer (confirmed working):

```
void * run(void * ar){

         char ** args= (char **) ar;
        matchingLines(args[0], args[1])  ;
        return NULL;
}
```

```
bool matchingLinesMany(const char* string, const char** filenames, int numfiles) {

        int i = 0;
        char ** args;
        pthread_t tid[numfiles];
        for(i; i < numfiles; i++){
                args = (char **) malloc(2*sizeof(char **));
                args[0] = string;
                args[1]= filenames[i];
                pthread_create(&tid[i], NULL, run, (void *) args);
        }

        i = 0;
        for(; i < numfiles; i++){
                pthread_join(tid[i], NULL);
        }
}
```

^^ Does not meet requirement to return true/false based upon any files resulting in errors. [BM]

D) Implement a function grepSearchMany which takes a string and an array of filenames (and its size) and prints (to stdout) all the lines in all the files which contain the string. All the specified files should be processed concurrently. If any of the files result in an error, then grepSearchMany will return false otherwise return true. Hint: You should call the function from part B. For this part, you can assume it exists and works. [6 marks]

bool grepSearchMany(const char* string, const char** filenames, int numfiles) {

```
bool grepSearchMany(const char* string, const char** filenames, int numfiles) {
        int childPid, status, pids[numfiles], pipefds[numfiles][2];

        for (int i = 0; i < numfiles; i++) {
                pipe(pipefds[i]);
                pids[i] = fork();

                if (pids[i] != 0) {
                        //parent
                        dup(pipefds[i][1]);
                        close(pipefds[i][0]);
                } else {
                        //child
                        close(STDIN_FILENO);
                        dup(pipefds[i][0]);
                        close(pipefds[i][1]);
                        if (grepSearch(string, filenames[i])) {
                                exit(0);
                        } else {
                                exit(2);
                        }
                }
        }

        bool success = true;
        for (int i = 0; i < numfiles; i++) {
                childPid = wait(&status);
                if (WEXITSTATUS(status) == 2) {
                        success = false;
                }
        }
        return success;
}
```

^^ int pids[numfiles] is not valid C, since numfiles is a variable. int *pids = malloc(sizeof(int) * numfiles) is (with a corresponding free at the end). [BM]

E) Write a program which takes the following arguments:
[6 marks]

- The character M or G.
- a string
- a sequence of filenames to process.

It will output all the lines in any of the files which contain the string. It will do this by calling
matchingLinesMany (if the character is 'M') or grepSearchMany (if the character is 'G'). For example:

./a.out M fish jungle ocean aquarium

Would search the files jungle, ocean, aquarium for lines containing fish.
For this part, you may assume that matchingLinesMany() and grepSearchMany() exist and function correctly.

Condition	Exit status	Message (to stderr)
No errors	0	
Incorrect number of parame- ters or invalid character	1	Bad params.
matchingLinesMany() or grepSearchMany() returns false	2	Bad file.

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

```
int main(int argc, const char **argv) {
        if (argc < 4) {
                fprintf(stderr, "Bad Params.");
                return 1;
        }
        if (strcmp(argv[1], "M") == 0) {
                if(!matchingLinesMany(argv[2], argv + 3, argc - 3)) {
                        fprintf(stderr, "Bad File.");
                        return 2;
                }
        } else if(strcmp(argv[1], "G") == 0) {
                if(!grepSearchMany(argv[2], argv + 3, argc - 3)) {
                        fprintf(stderr, "Bad File.");
                        return 2;
                }
        } else {
                fprintf(stderr, "Bad Params.");
                return 1;
        }

        return 0;
}
```