COMP 304 Project 1: Seashell

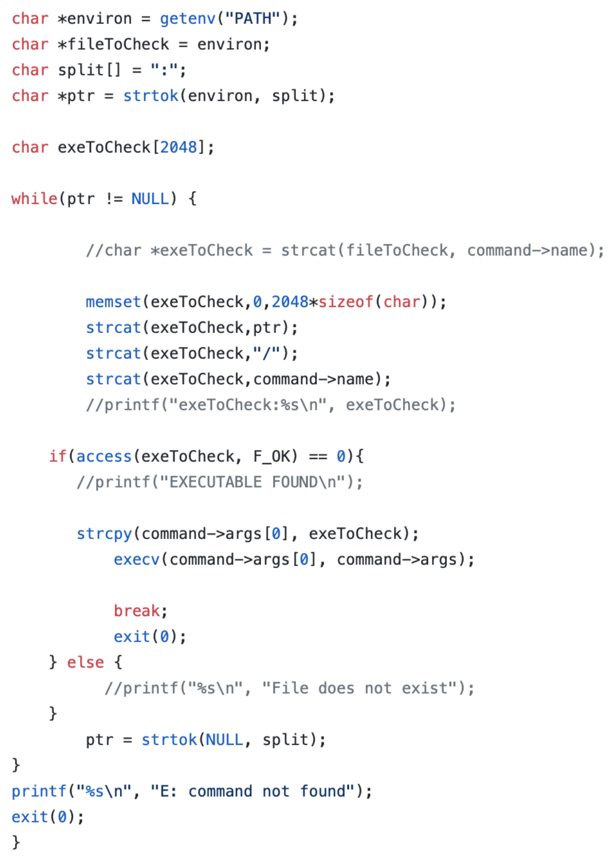
Ahmet Akkoç 64741, Kutluhan Palalıoğlu 64795

How to Compile and Run:

```
> gcc seashell.c -o seashell
> ./seashell
```

Explanation

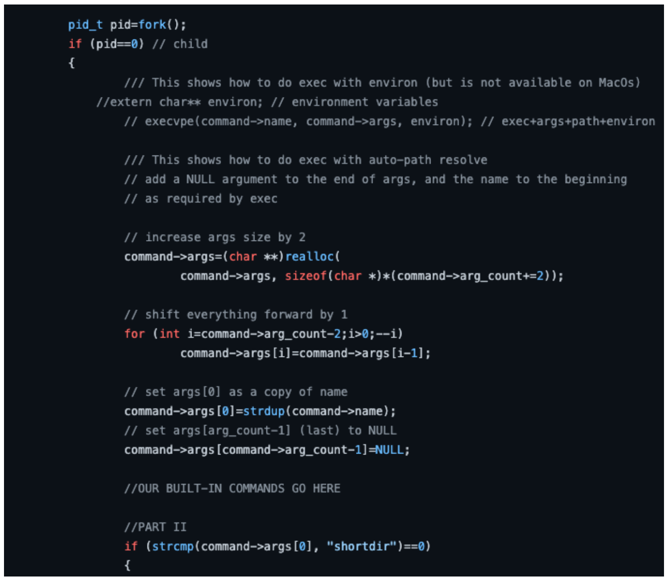
Part 1



```
char *environ = getenv("PATH");
char *fileToCheck = environ;
char split[] = ":";
char *ptr = strtok(environ, split);

char exeToCheck[2048];

while(ptr != NULL) {

        //char *exeToCheck = strcat(fileToCheck, command->name);

        memset(exeToCheck,0,2048*sizeof(char));
        strcat(exeToCheck,ptr);
        strcat(exeToCheck,"/");
        strcat(exeToCheck,command->name);
        //printf("exeToCheck:%s\n", exeToCheck);

    if(access(exeToCheck, F_OK) == 0){
       //printf("EXECUTABLE FOUND\n");

       strcpy(command->args[0], exeToCheck);
           execv(command->args[0], command->args);

           break;
           exit(0);
    } else {
          //printf("%s\n", "File does not exist");
    }
        ptr = strtok(NULL, split);
}
printf("%s\n", "E: command not found");
exit(0);
}
```

For this part, we firstly got the environment of the path and tokenized it with ":" by using strtok function in string.h library. Then for each token we checked whether the executable that we are looking for exists and we have access to it. Finally we executed the command with the execv function.

Since we started the assignment early, you might notice traces of a solution for the original part I in our code, including double bangs (!!'s) and a history function. After the revised assignment, we decided to recycle this code for part VI's myfavorite command.

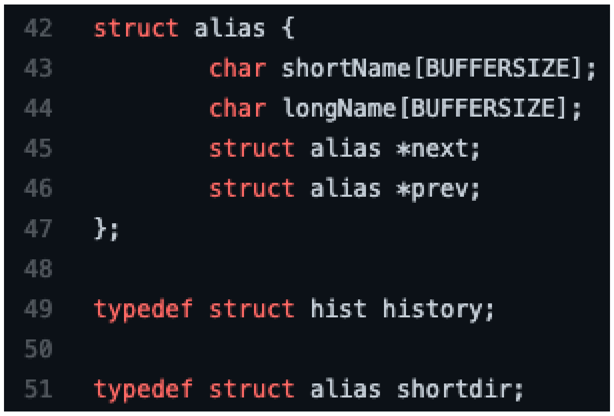


```c
        pid_t pid=fork();
        if (pid==0) // child
        {
                /// This shows how to do exec with environ (but is not available on MacOs)
            //extern char** environ; // environment variables
                // execvpe(command->name, command->args, environ); // exec+args+path+environ

                /// This shows how to do exec with auto-path resolve
                // add a NULL argument to the end of args, and the name to the beginning
                // as required by exec

                // increase args size by 2
                command->args=(char **)realloc(
                        command->args, sizeof(char *)*(command->arg_count+=2));

                // shift everything forward by 1
                for (int i=command->arg_count-2;i>0;--i)
                        command->args[i]=command->args[i-1];

                // set args[0] as a copy of name
                command->args[0]=strdup(command->name);
                // set args[arg_count-1] (last) to NULL
                command->args[command->arg_count-1]=NULL;

                //OUR BUILT-IN COMMANDS GO HERE

                //PART II
                if (strcmp(command->args[0], "shortdir")==0)
                {
```

Part 2

For parts 2-6, we decided to code our built-in commands to execute after a **process fork** so that execution errors would not interrupt shell execution. Note the //OUR BUILT-IN COMMANDS GO HERE line in the code snippet on the left.

In part 2, we represented our shortdir aliases as `struct alias`es, typedef'ed as `shortdir`. Our shortdir aliases were stored in a doubly linked list of shortdirs, which gave us O(n) time complexity on insertions, deletions and accesses. The implementations for shortdir set jump clear del and list can be found between lines 492-593.



```c
struct alias {
        char shortName[BUFFERSIZE];
        char longName[BUFFERSIZE];
        struct alias *next;
        struct alias *prev;
};

typedef struct hist history;

typedef struct alias shortdir;
```

To preserve aliases from session to session, we save them into `$HOME/aliases.txt`. At the start of a new session, seashell attempts to `load_aliases` from any previous sessions and will `save_aliases` at regular intervals.

Part 3

For this part, we used a file pointer to read the file with the fopen() function. The outer loop loops until we reach EOF. In the inner loop, we tokenize the line word by word and put the tokens(words) in a new string variable that is called “lineAbouttaBePrinted”.

```
//checking whether this token is what we are looking for
if(strcasecmp(token, word) == 0) {
        //Turn the string into red
        if(strcmp(color, "r") == 0) {
                char red[512] = "\e[31m\e[5m\e[1m";
                strcat(red, token);
                strcat(red, "\033[1m\033[0m");
                token = red;
                stringsOfColor = 1;
        }
```

---> Here we are checking whether the current token is what we are looking for. If it is, we highlight that token (red for this conditional statement) and the stringsOfColor variable is set to 1 which triggers the print statement for this line. Without stringsOfColor, we would only reconstruct the line on a new string that wouldn’t be printed.

Part 4

In this part, firstly, we took the inputs and matched them with hour, minute. Secondly, after getting this hour and minute information, we wrote them in a file with crontab format. Finally we executed the command as follows:

```
fprintf(fptr, "%d %d * * * DISPLAY=:0.0 /usr/bin/rhythmbox-client --play %s\n", min, hour, filename);

fclose(fptr);

//3. exec to read crontab

//$crontab alarm.txt
//crontab alarm.txt
char *crontabexec[2] = {"crontab", alarmfile};
execvp(crontabexec[0], crontabexec);
exit(0);
```

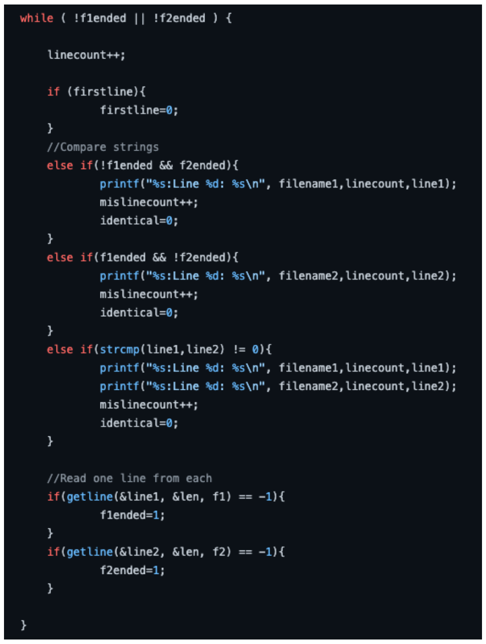
Our alarm job (alarmfile) is past to crontab via `$HOME/alarm.txt`

Part 5

```
if(command->arg_count == 4) mode = 0;
else if(command->arg_count == 5){
        char c;
        sscanf(command->args[1], "-%c", &c);

        mode = ((c=='a')?0:1);
        //printf("Mode: %d\n",mode);
}
else{
        printf("E: Incorrect number of arguments for kdiff (2 or 3) \n");
        return EXIT;
}
```

---> Here we check if the -a or -b option is specified, assuming -a if neither is provided. This mode determines which process we are going to follow “-a” or “-b”.

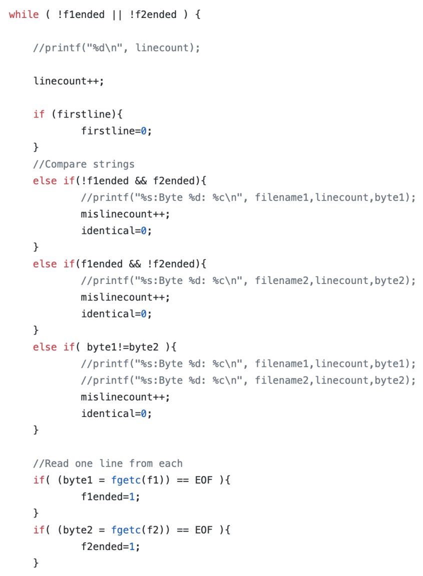
Then we match the command arguments with our file variables.



```
while ( !f1ended || !f2ended ) {

    linecount++;

    if (firstline){
            firstline=0;
    }
    //Compare strings
    else if(!f1ended && f2ended){
            printf("%s:Line %d: %s\n", filename1,linecount,line1);
            mislinecount++;
            identical=0;
    }
    else if(f1ended && !f2ended){
            printf("%s:Line %d: %s\n", filename2,linecount,line2);
            mislinecount++;
            identical=0;
    }
    else if(strcmp(line1,line2) != 0){
            printf("%s:Line %d: %s\n", filename1,linecount,line1);
            printf("%s:Line %d: %s\n", filename2,linecount,line2);
            mislinecount++;
            identical=0;
    }

    //Read one line from each
    if(getline(&line1, &len, f1) == -1){
            f1ended=1;
    }
    if(getline(&line2, &len, f2) == -1){
            f2ended=1;
    }

}
```

---> In part “A” of this question, we check if the lines are identical. If we already reached the end of one of the files, the extra lines of the other file will continue to be printed.

If no differences are found we print `"The two files are identical\n\n"`



```
while ( !f1ended || !f2ended ) {

    //printf("%d\n", linecount);

    linecount++;

    if (firstline){
            firstline=0;
    }
    //Compare strings
    else if(!f1ended && f2ended){
            //printf("%s:Byte %d: %c\n", filename1,linecount,byte1);
            mislinecount++;
            identical=0;
    }
    else if(f1ended && !f2ended){
            //printf("%s:Byte %d: %c\n", filename2,linecount,byte2);
            mislinecount++;
            identical=0;
    }
    else if( byte1!=byte2 ){
            //printf("%s:Byte %d: %c\n", filename1,linecount,byte1);
            //printf("%s:Byte %d: %c\n", filename2,linecount,byte2);
            mislinecount++;
            identical=0;
    }

    //Read one line from each
    if( (byte1 = fgetc(f1)) == EOF ){
            f1ended=1;
    }
    if( (byte2 = fgetc(f2)) == EOF ){
            f2ended=1;
    }
```

In part “B” of this question we do the same thing as part “A” <----- however we do the comparison char by char using `fgetc` and take their byte value into account.

If no differences are found we print `"The two files are identical\n\n"`

Part 6

Since we started the assignment early, you might notice traces of a solution for the original part I including double bangs (!!’s) and a history function. After the revised assignment, we decided to recycle this code for part VI’s myfavorite command.

The myfavorite command tells the user their most frequently used command.

```
//Step 1: Loop over commands
//If not checked mark and begin counting
//If checked continue
int i,j;
for(i = h->length - 1; i >= 0 ;i--){
        if(checked[i]) continue;
        //Not checked before, checking now
        count[i] = checked[i] = 1;
        for(j = i - 1; j >= 0 ;j--){
                if(checked[j]) continue;
                //Not matched before, attempting to match now
                else if(strcmp(h->commands[i],h->commands[j])==0){
                        count[i] += checked[j] = 1;
                }
        }
}
```

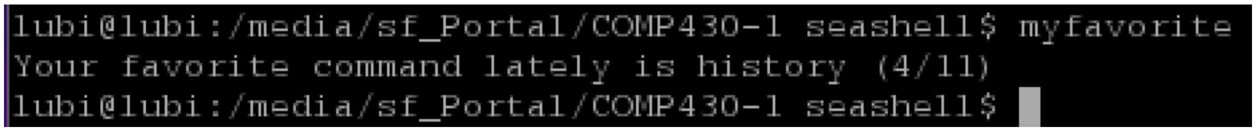
---> In this part, until there is no unchecked command left, we put a check on them and add one to the corresponding command count value. For a visualization of this process, see the comment at the bottom of Step 3.

```
//Step 2: Loop over count to find largest count
int fav=-1, favcount=-1;
for(i = h->length - 1; i >= 0 ;i--){
        if(count[i] > favcount){
                favcount = count[i];
                fav = i;
        }
}
```

---> This is a loop to find the command with the highest count.

```
//Step 3: Return corresponding string with largest count
printf("Your favorite command lately is %s (%d/%d)\n", h->commands[fav], favcount,h->length);

   /* checked   count
       a 1        2
       b 1        3
       c 1        1
       d 1        1
       b 1        0
       a 1        0
       b 1        0
   */
```

---> After getting all the info we print the *fav* command, which is the most used command in the last 20 commands.



```
lubi@lubi:/media/sf_Portal/COMP430-1 seashell$ myfavorite
Your favorite command lately is history (4/11)
lubi@lubi:/media/sf_Portal/COMP430-1 seashell$
```