Lap1

Name: Mohamed Ahmed Aly Riyad

Id: 20011457

Problem statement:

It is required to implement a Unix shell program. A shell is simply a program that conveniently allows you to run other programs. Read up on your favorite shell to see what it does.

Your shell must support the following commands:

The internal shell command "exit" which terminates the shell

Concepts: shell commands, exiting the shell.

System calls: exit()

A command with no arguments

Example: ls, cp, rm …etc

Details: Your shell must block until the command completes and, if the return code is abnormal, print out a message to that effect.

Concepts: Forking a child process, waiting for it to complete and synchronous execution.

System calls: fork(), execvp(), exit(), waitpid()

A command with arguments

Example: ls –l

Details: Argument 0 is the name of the command.

Concepts: Command-line parameters.

A command, with or without arguments, executed in the background using &.

Example: firefox &

Details: In this case, your shell must execute the command and return immediately, not blocking until the command finishes.

Concepts: Background execution, signals, signal handlers, processes and asynchronous execution.

Requirements: You have to show that the opened process will be nested as a child process to the shell program via opening the task manager found in the operating system like the one shown in figure 1. Additionally you have to write in a log file (basic text file) when a child process is terminated (main application will be interrupted by a SIGCHLD signal). So you have to implement an interrupt handler to handle this interrupt and do the corresponding action to it.

Shell builtin commands

Commands: cd & echo

Details: for the case of:

cd: Cover all the following cases (assume no spaces in path):

cd

cd ~

cd ..

cd absolute_path

cd relative_path_to_current_working_directory

echo: Prints the input after evaluating all expressions (assume input to echo must be within double quotations).

echo "wow" => wow

export x=5

echo "Hello $x" => Hello 5

Expression evaluation

Commands: export

Details: Set values to variables and print variables values. No mathematical operations is needed.

Export Details: Accept input of two forms, either a string without spaces, or a full string inside double quotations.

Example:

export x=-l

ls $x => Will perform ls -l

export y="Hello world"

echo "$y" => Hello world

A description of the overall organization of your code

1.taking input from user as string

2.converting string to many strings(commands)

3.store values after export

4.evaluate values after $

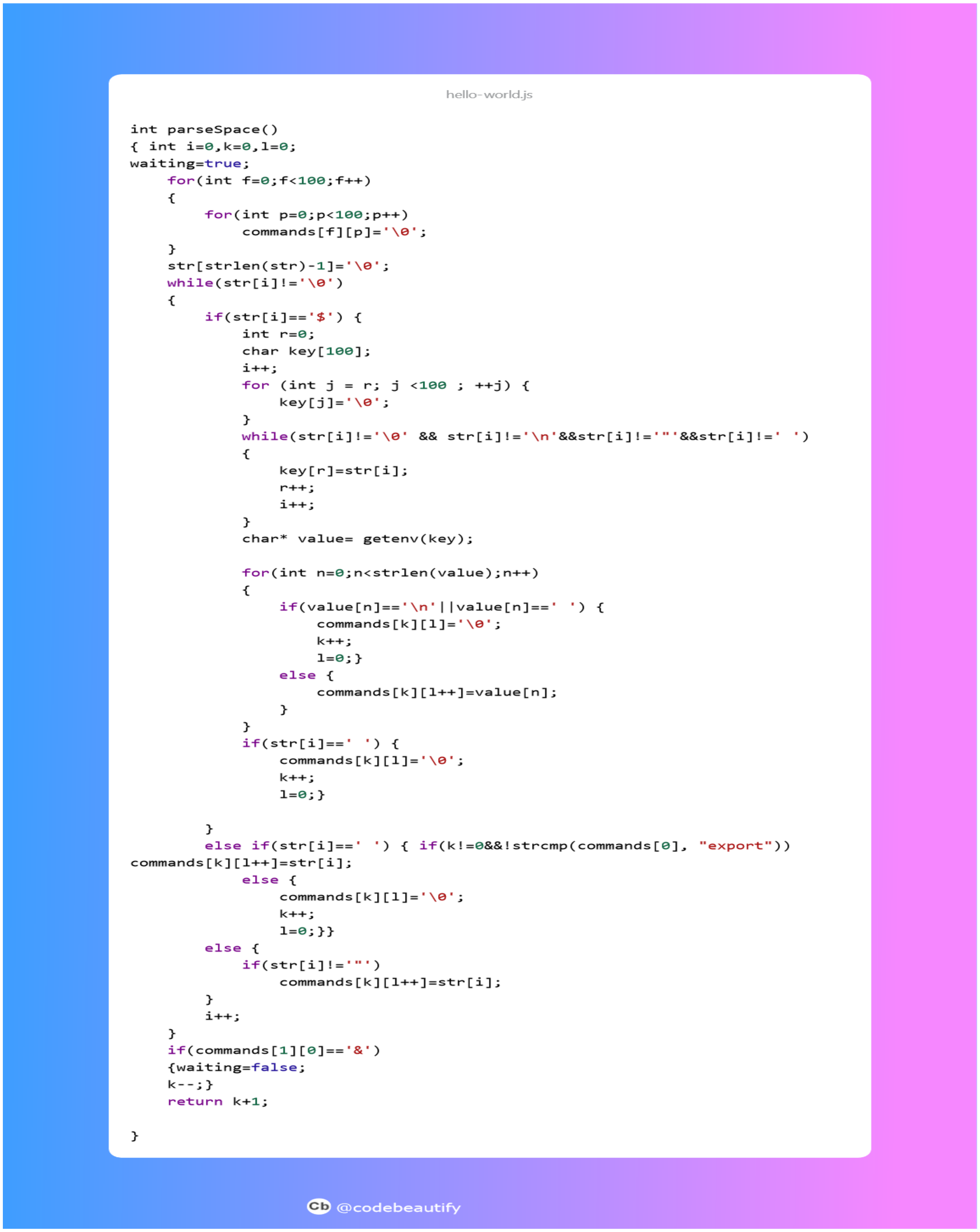
5.function of builtin commands(echo-export-cd)

6.function of other commands

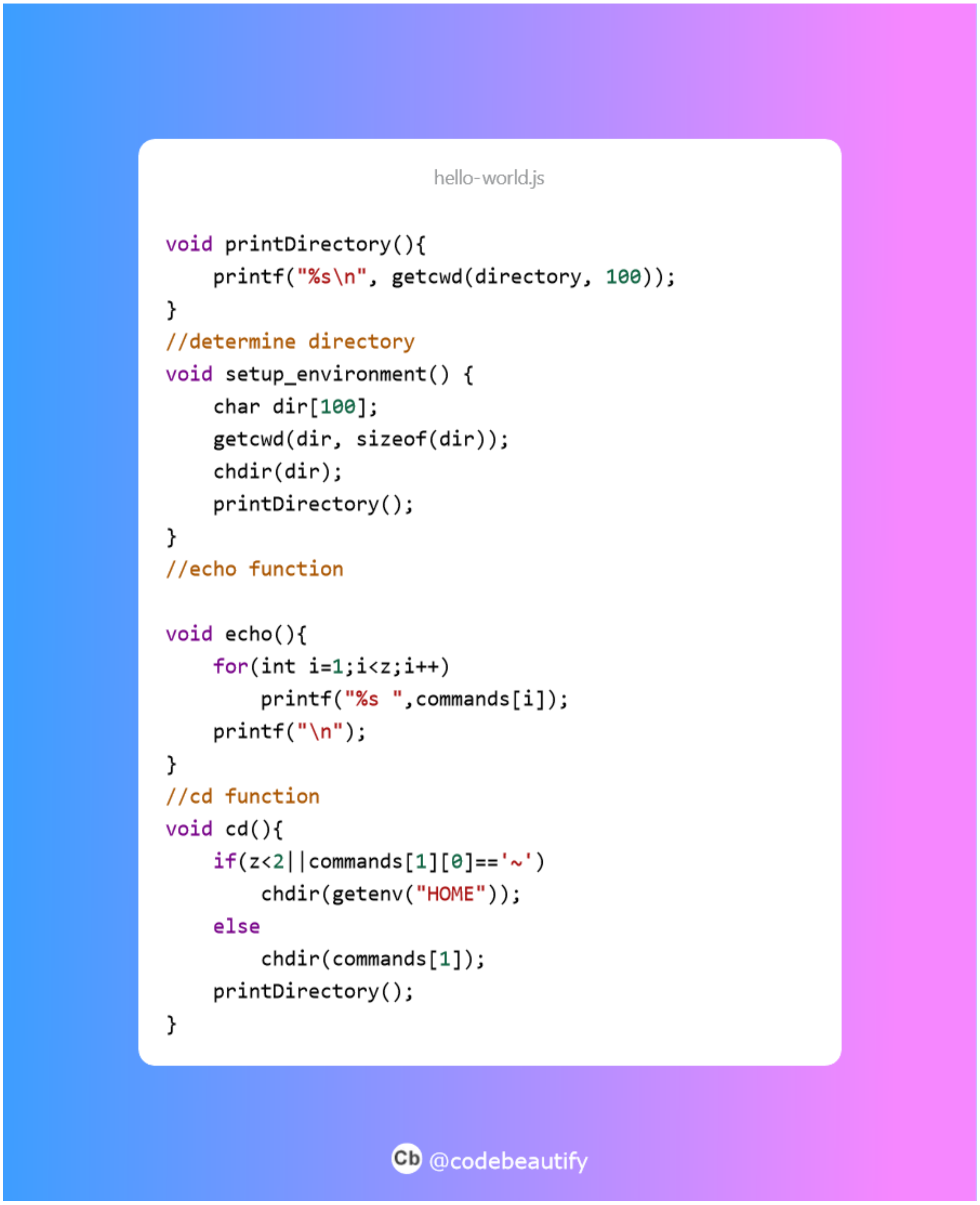
7.killing zombie processe

8.log child termination to text file

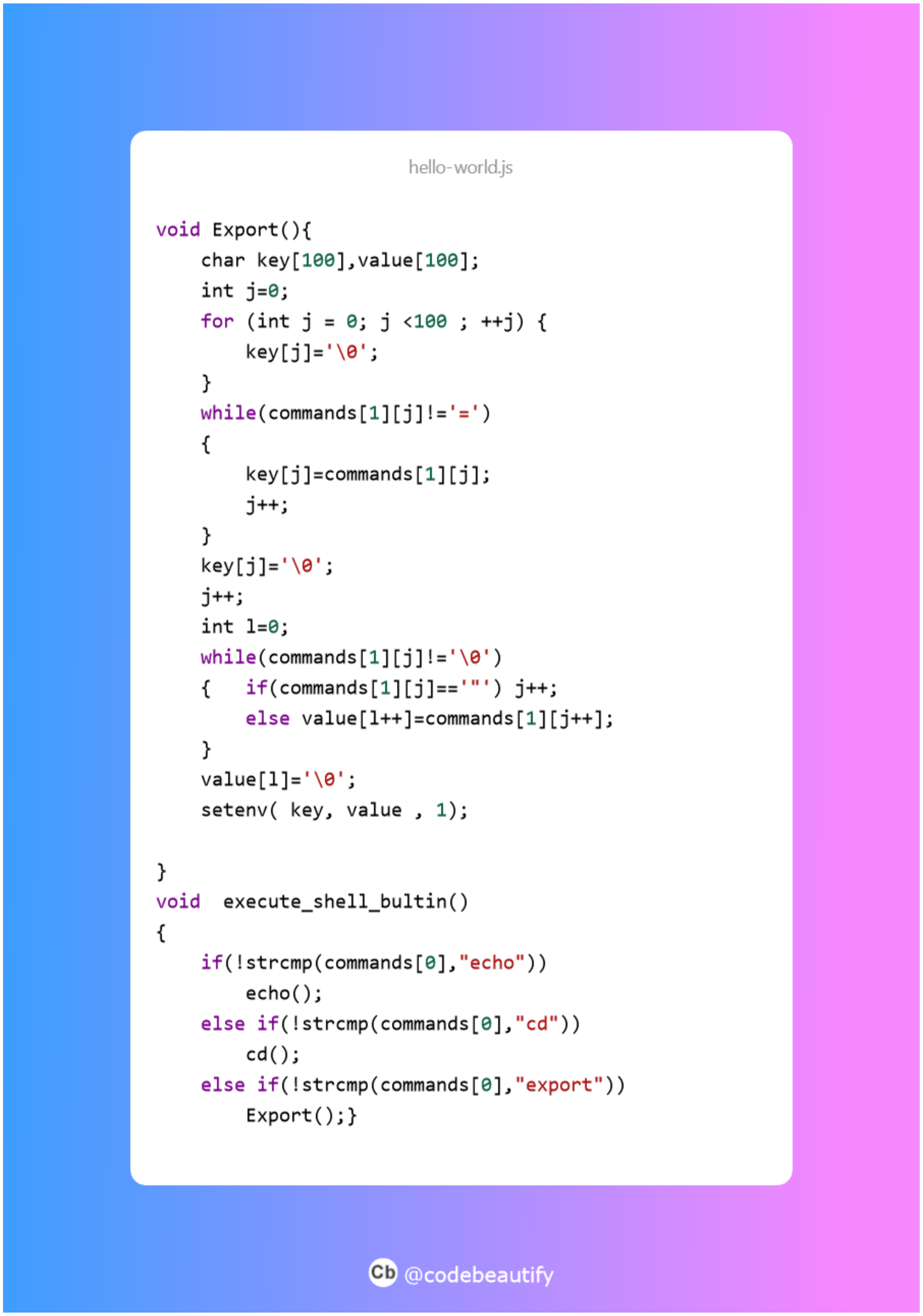
The major function:



```
int parseSpace()
{ int i=0,k=0,l=0;
waiting=true;
    for(int f=0;f<100;f++)
    {
        for(int p=0;p<100;p++)
            commands[f][p]='\0';
    }
    str[strlen(str)-1]='\0';
    while(str[i]!='\0')
    {
        if(str[i]=='$') {
            int r=0;
            char key[100];
            i++;
            for (int j = r; j <100 ; ++j) {
                key[j]='\0';
            }
            while(str[i]!='\0' && str[i]!='\n'&&str[i]!='"'&&str[i]!=' ')
            {
                key[r]=str[i];
                r++;
                i++;
            }
            char* value= getenv(key);

            for(int n=0;n<strlen(value);n++)
            {
                if(value[n]=='\n'||value[n]==' ') {
                    commands[k][l]='\0';
                    k++;
                    l=0;}
                else {
                    commands[k][l++]=value[n];
                }
            }
            if(str[i]==' ') {
                commands[k][l]='\0';
                k++;
                l=0;}

        }
        else if(str[i]==' ') { if(k!=0&&!strcmp(commands[0], "export"))
commands[k][l++]=str[i];
            else {
                commands[k][l]='\0';
                k++;
                l=0;}}
        else {
            if(str[i]!='"')
                commands[k][l++]=str[i];
        }
        i++;
    }
    if(commands[1][0]=='&')
    {waiting=false;
    k--;}
    return k+1;

}
```

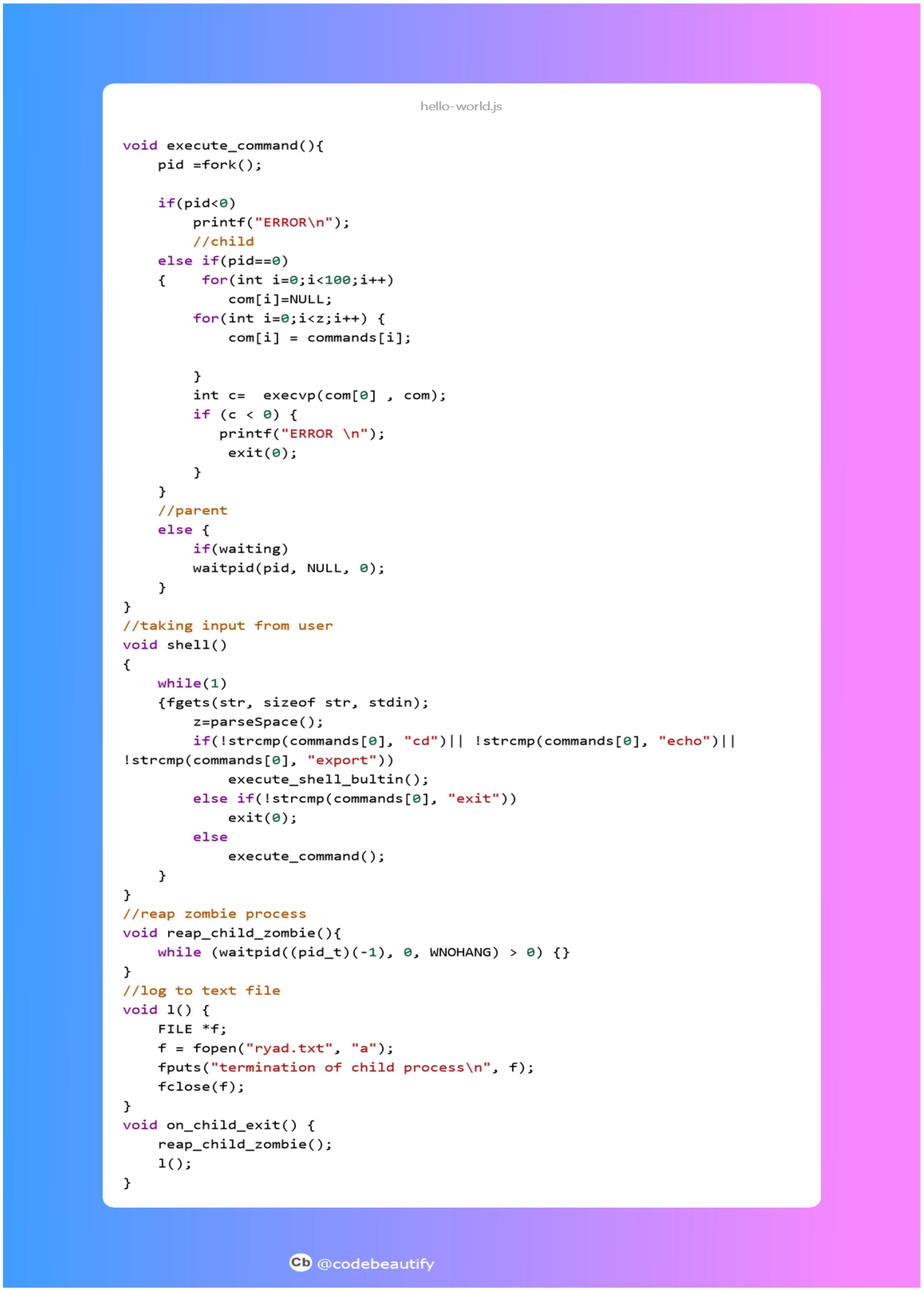


```
void printDirectory(){
    printf("%s\n", getcwd(directory, 100));
}
//determine directory
void setup_environment() {
    char dir[100];
    getcwd(dir, sizeof(dir));
    chdir(dir);
    printDirectory();
}
//echo function

void echo(){
    for(int i=1;i<z;i++)
        printf("%s ",commands[i]);
    printf("\n");
}
//cd function
void cd(){
    if(z<2||commands[1][0]=='~')
        chdir(getenv("HOME"));
    else
        chdir(commands[1]);
    printDirectory();
}
```


hello-world.js

```
void Export(){
    char key[100],value[100];
    int j=0;
    for (int j = 0; j <100 ; ++j) {
        key[j]='\0';
    }
    while(commands[1][j]!='=')
    {
        key[j]=commands[1][j];
        j++;
    }
    key[j]='\0';
    j++;
    int l=0;
    while(commands[1][j]!='\0')
    {   if(commands[1][j]=='"') j++;
        else value[l++]=commands[1][j++];
    }
    value[l]='\0';
    setenv( key, value , 1);

}
void  execute_shell_bultin()
{
    if(!strcmp(commands[0],"echo"))
        echo();
    else if(!strcmp(commands[0],"cd"))
        cd();
    else if(!strcmp(commands[0],"export"))
        Export();}
```

@codebeautify

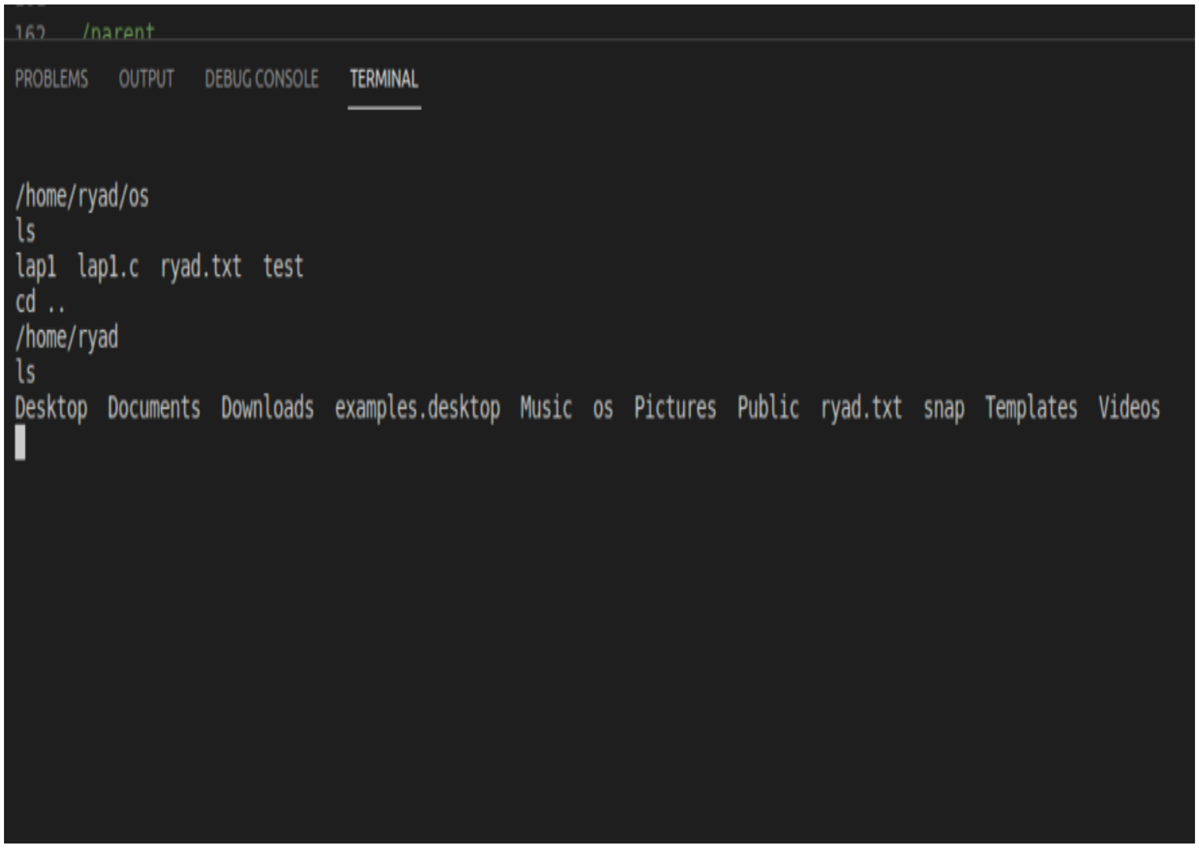


hello-world.js

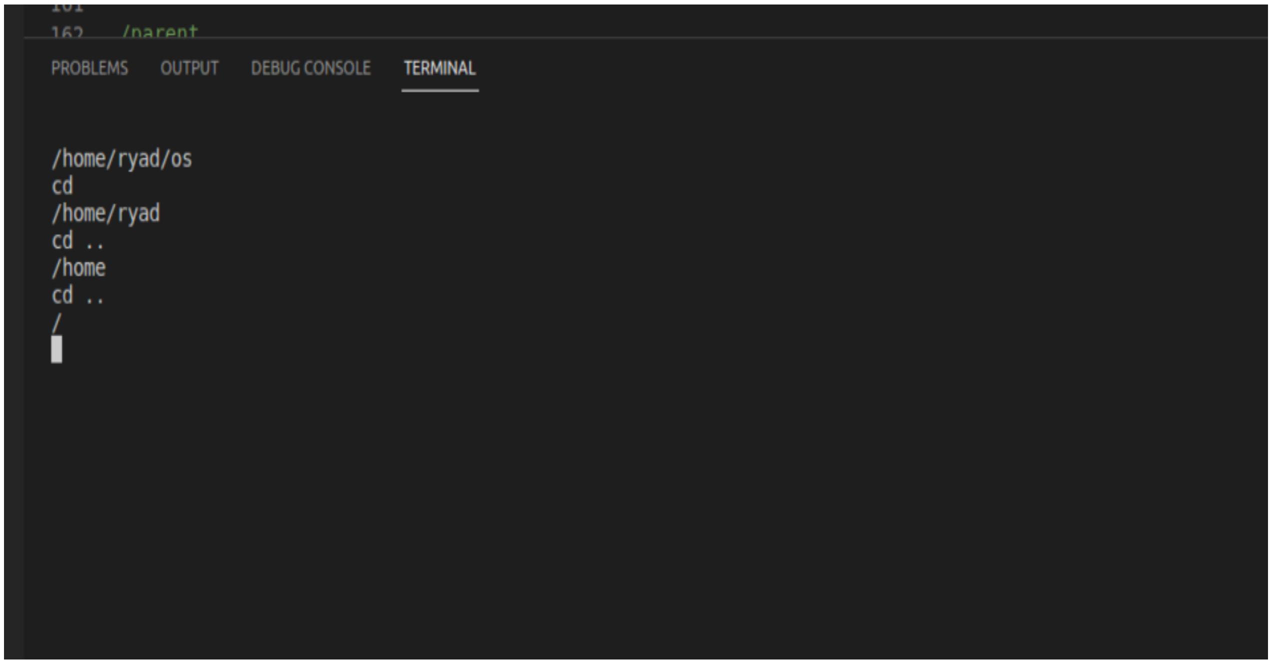
```
void execute_command(){
    pid =fork();

    if(pid<0)
        printf("ERROR\n");
        //child
    else if(pid==0)
    {    for(int i=0;i<100;i++)
            com[i]=NULL;
        for(int i=0;i<z;i++) {
            com[i] = commands[i];

        }
        int c=  execvp(com[0] , com);
        if (c < 0) {
           printf("ERROR \n");
            exit(0);
        }
    }
    //parent
    else {
        if(waiting)
        waitpid(pid, NULL, 0);
    }
}
//taking input from user
void shell()
{
    while(1)
    {fgets(str, sizeof str, stdin);
        z=parseSpace();
        if(!strcmp(commands[0], "cd")|| !strcmp(commands[0], "echo")||
!strcmp(commands[0], "export"))
            execute_shell_bultin();
        else if(!strcmp(commands[0], "exit"))
            exit(0);
        else
            execute_command();
    }
}
//reap zombie process
void reap_child_zombie(){
    while (waitpid((pid_t)(-1), 0, WNOHANG) > 0) {}
}
//log to text file
void l() {
    FILE *f;
    f = fopen("ryad.txt", "a");
    fputs("termination of child process\n", f);
    fclose(f);
}
void on_child_exit() {
    reap_child_zombie();
    l();
}
```

@codebeautify

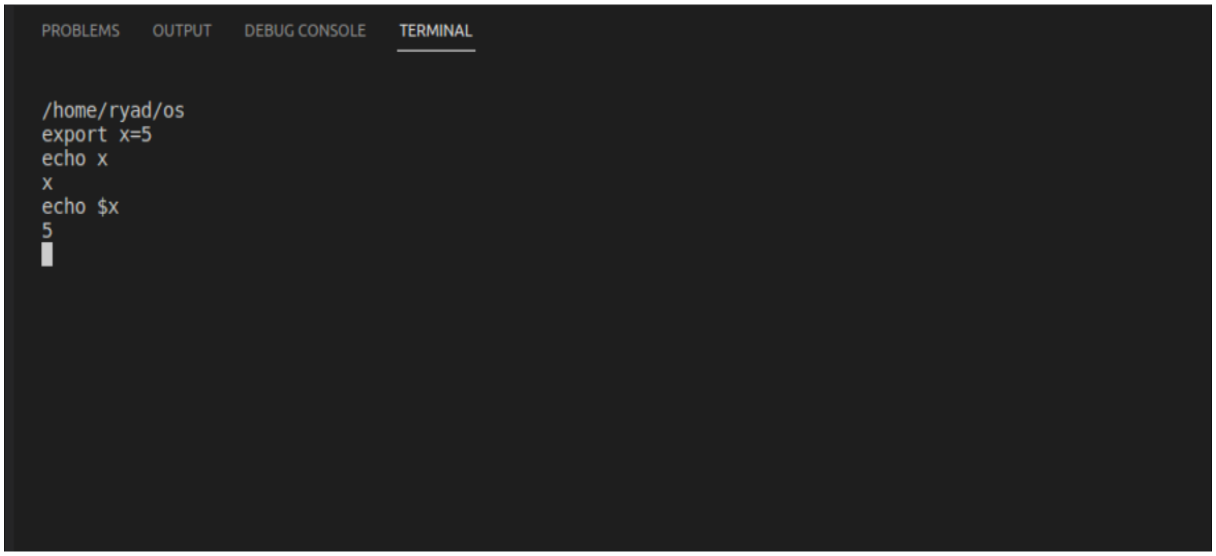
Sample runs:



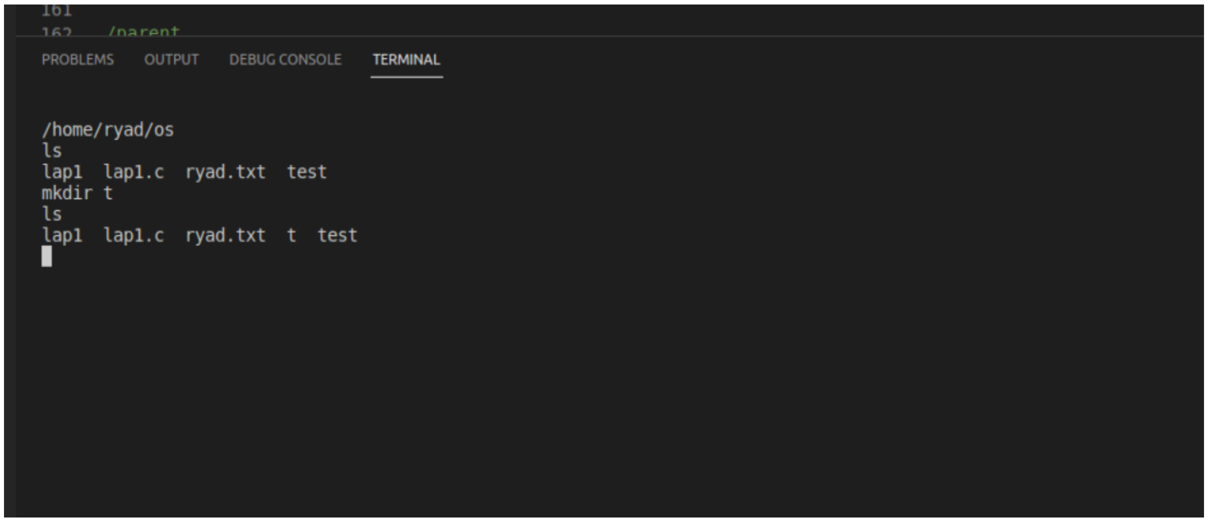
```
/home/ryad/os
ls
lap1  lap1.c  ryad.txt  test
cd ..
/home/ryad
ls
Desktop  Documents  Downloads  examples.desktop  Music  os  Pictures  Public  ryad.txt  snap  Templates  Videos
```



```
/home/ryad/os
cd
/home/ryad
cd ..
/home
cd ..
/
```

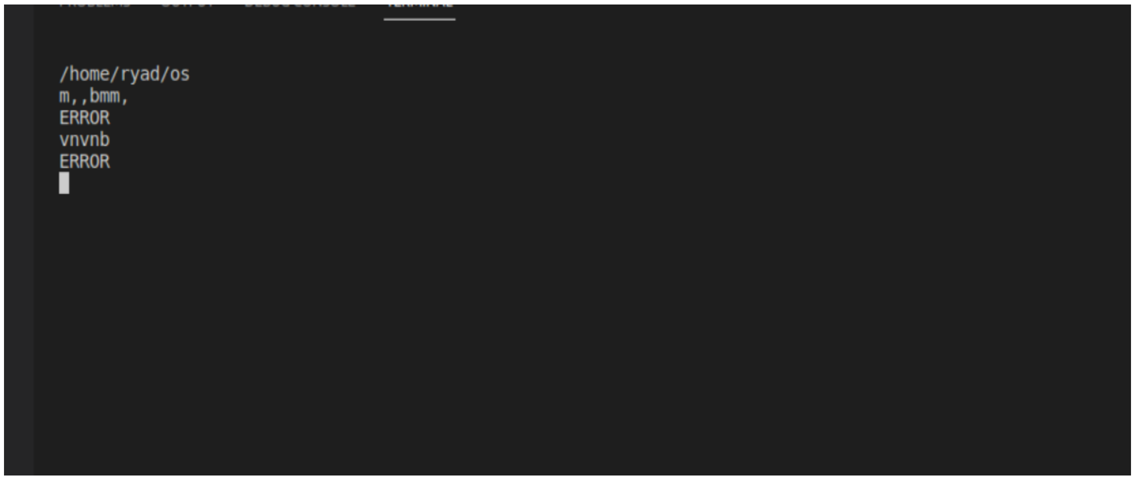


```
/home/ryad/os
export x=5
echo x
x
echo $x
5
```



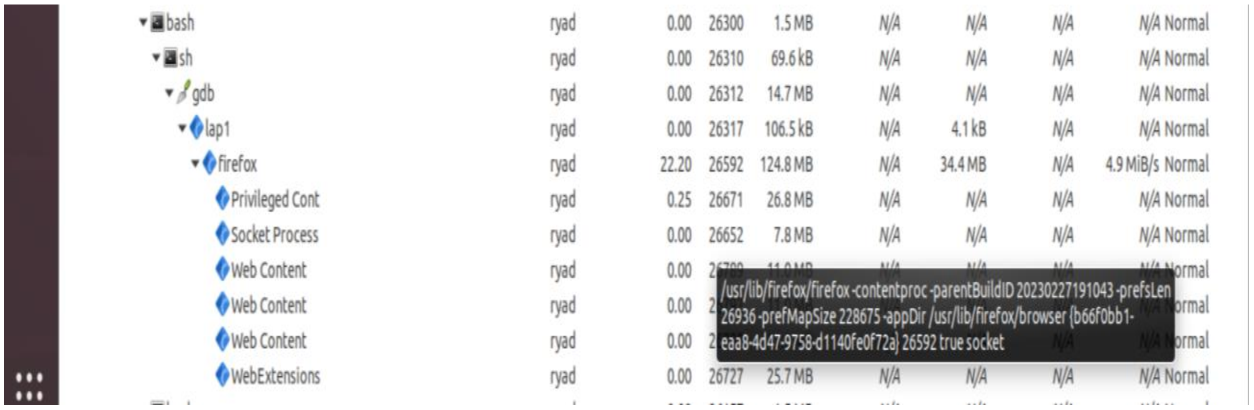
```
/parent
```

```
/home/ryad/os
ls
lap1  lap1.c  ryad.txt  test
mkdir t
ls
lap1  lap1.c  ryad.txt  t  test
```



```
/home/ryad/os
m,,bmm,
ERROR
vnvnb
ERROR
```

screenshots for the processes hierarchy in KSysguard (or any similar package) during the execution of your shell program



```
▾ bash                         ryad    0.00  26300    1.5 MB   N/A      N/A   N/A       N/A Normal
  ▾ sh                         ryad    0.00  26310   69.6 kB   N/A      N/A   N/A       N/A Normal
    ▾ gdb                      ryad    0.00  26312   14.7 MB   N/A      N/A   N/A       N/A Normal
      ▾ lap1                   ryad    0.00  26317  106.5 kB   N/A   4.1 kB   N/A       N/A Normal
        ▾ firefox              ryad   22.20  26592  124.8 MB   N/A  34.4 MB   N/A  4.9 MiB/s Normal
            Privileged Cont    ryad    0.25  26671   26.8 MB   N/A      N/A   N/A       N/A Normal
            Socket Process     ryad    0.00  26652    7.8 MB   N/A      N/A   N/A       N/A Normal
            Web Content        ryad    0.00  26789   11.0 MB
            Web Content        ryad    0.00
            Web Content        ryad    0.00
            WebExtensions      ryad    0.00  26727   25.7 MB   N/A      N/A   N/A       N/A Normal
```

```
/usr/lib/firefox/firefox -contentproc -parentBuildID 20230227191043 -prefsLen
26936 -prefMapSize 228675 -appDir /usr/lib/firefox/browser {b66f0bb1-
eaa8-4d47-9758-d1140fe0f72a} 26592 true socket
```