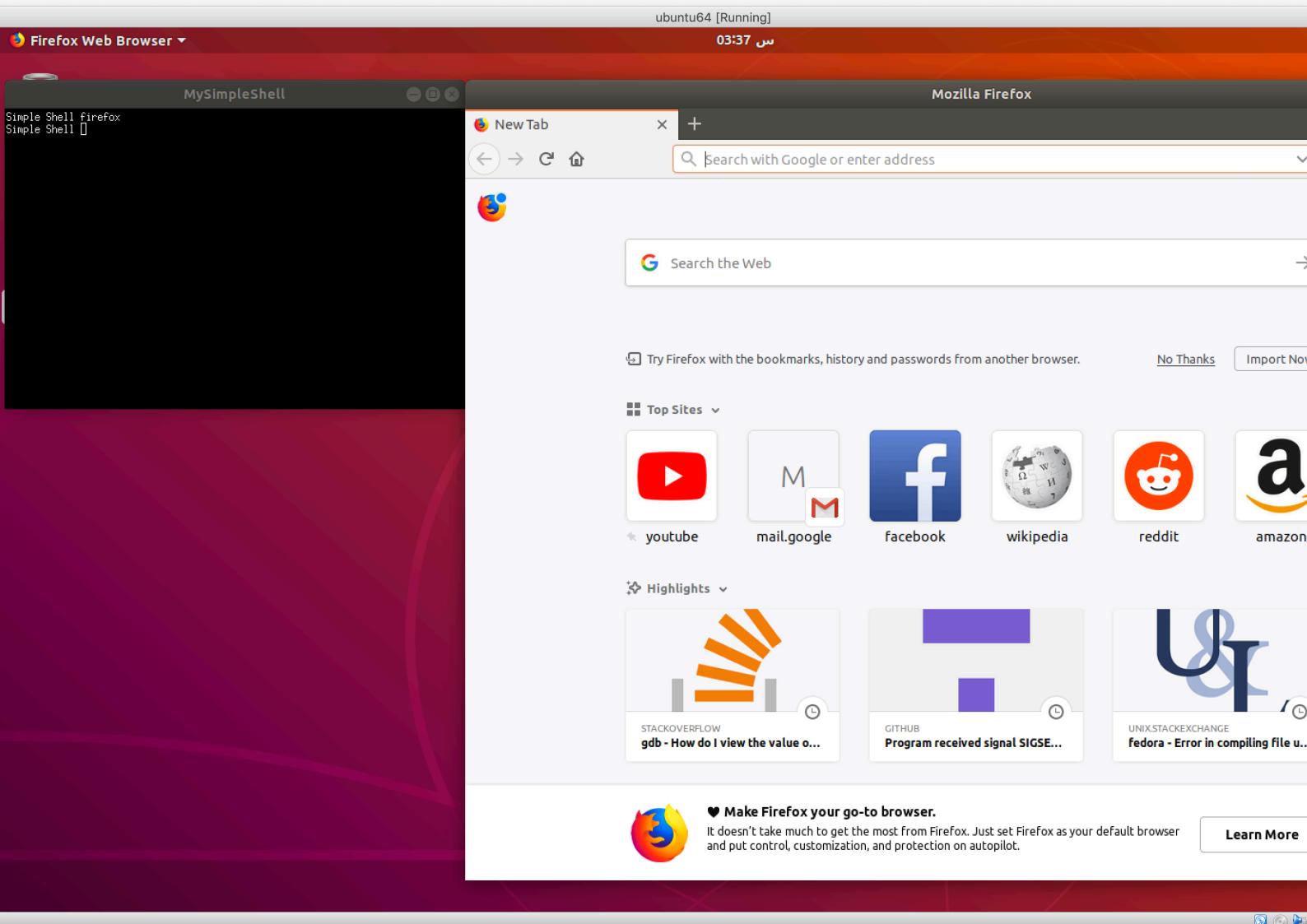


Karim Fathy 4269

Simple Shell

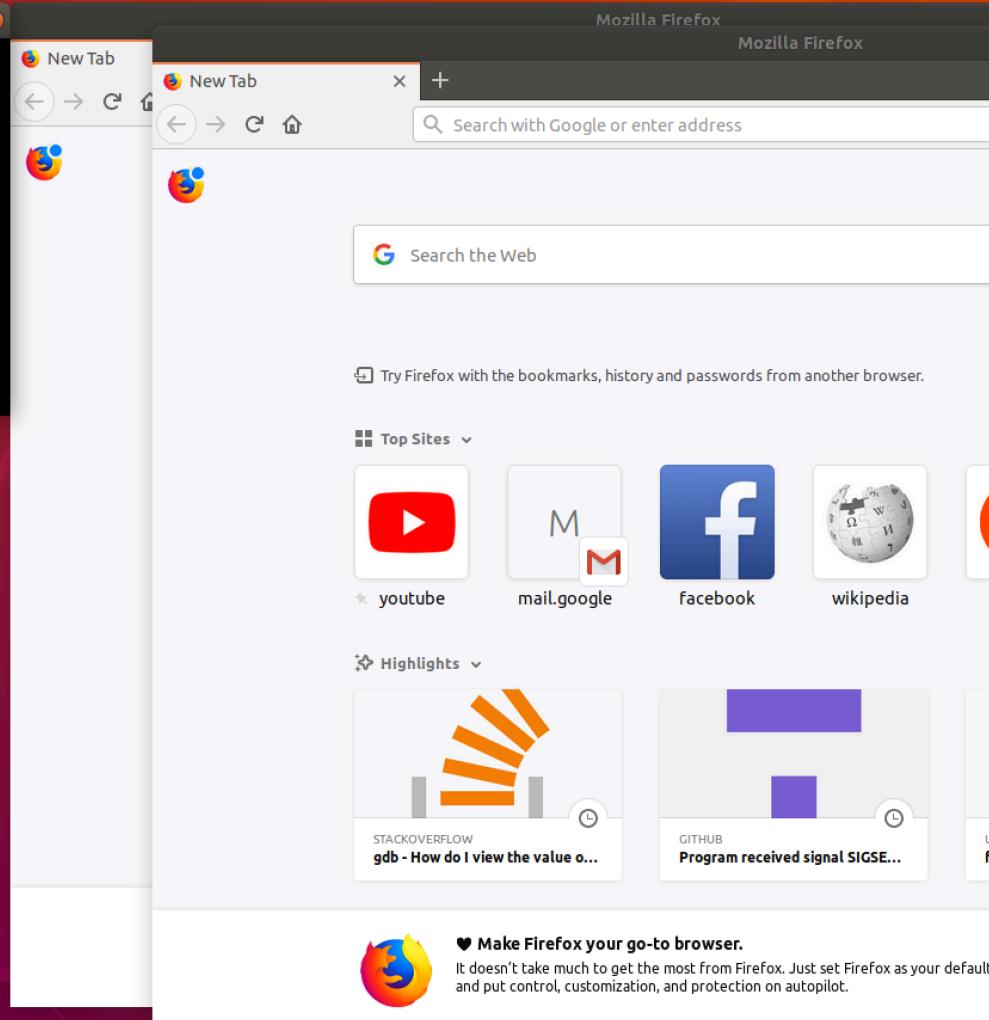
Sample runs:



s XTerm ▾

MySimpleShell

```
Simple Shell firefox
Simple Shell firefox &
Simple Shell
```



XTerm

MySimpleShell

```
Simple Shell firefox &
Simple Shell ls
bin main.c           MySimpleShell.depend obj
log MySimpleShell.cbp MySimpleShell.layout
Simple Shell pwd
/home/kimo/Desktop/MySimpleShell
Simple Shell ls -l
total 28
drwxr-xr-x 3 kimo kimo 4096 26 01:52 bin
-rw-r--r-- 1 kimo kimo  75 27 03:47 log
-rw-r--r-- 1 kimo kimo 1848 27 03:43 main.c
-rw-r--r-- 1 kimo kimo  804 26 01:51 MySimpleShell.cbp
-rw-r--r-- 1 kimo kimo 159 27 03:44 MySimpleShell.depend
-rw-r--r-- 1 kimo kimo  359 27 03:37 MySimpleShell.layout
drwxr-xr-x 3 kimo kimo 4096 26 01:52 obj
Simple Shell
```

Ubuntu Software

Mozilla Firefox Mozilla Firefox

New Tab New Tab +

Search with Google or enter address

Try Firefox with the bookmarks, history and passwords from another Firefox profile.

Top Sites

- youtube
- M mail.google
- f facebook

Highlights

- STACKOVERFLOW How to use delay() function in c...
- GITHUB Program received sig

Browse with the best of 'em.
Prepare yourself for the fastest, smoothest, most reliable web browser.

ubuntu64 [Running]

03:53 س

main.c [MySimpleShell] - Code::Blocks 16.01

Mozilla Firefox

New Tab +

Search with Google or enter address

System Monitor

File View Settings Help

Process Table System Load

All Processes, Tree ▾

Name	Username	CPU %	Memory	Shared Mem
systemd	root	24%	٢٣٣٤ K	٦٨٨٨
gdm3	root	24%	١٢٧٢ K	٦١٣٨
gdm-session-wor	root	24%	١٨٣٨ K	٦٤٩٦
gdm-x-session	kimo	24%	٦٢٨ K	٥٣٢٨
gnome-session-binary	kimo	21%	٢٦٦٨ K	١١٤٣٢
gnome-shell	kimo	21%	٤٠٣٣١٦ K	٩٤٨٤
codeblocks	kimo	8%	٧٣٣١٦ K	٤٨٣٠٦
xterm	kimo	8%	٤٠٠٤ K	٦٢٠٦
cb_console_runner	kimo	8%	١٤٤ K	١٠٥٢
sh	kimo	8%	٦٨ K	٦١٢
MySimpleShell	kimo	8%	٦٠ K	٦٤٨
firefox	kimo	8%	١٢٠١٠٨ K	١٠٠٨
Web	kimo		٤٢٩٤٠ K	١٠١٢٢
Web	kimo		٣٩١٤٤ K	١٠٢٥٣
Web	kimo		٣٩٠٢٨ K	١٠٠٨
WebExtensions	kimo		٢٣٠١٢ K	٨٠٣٩
Web	kimo		١٦٧٠٦ K	٦٢٠٠
firefox	kimo		zombie	

18 processes CPU: 11% Memory: ١٤ GiB / ٣٩ GiB Swap: ٠ B / ٩٤٧٢ MiB

Browse with the best of 'em.
Prepare yourself for the fastest, smoothest, most reliable Firefox yet.

Learn More

Firefox logo

This part is responsible for forking new child process and distinguishing between a foreground and a background process. It is also responsible for executing of processes.

The screenshot shows the Code::Blocks IDE interface with a running application titled "MySimpleShell". The main window displays the following C code:

```
ubuntu64 [Running]
04:00 ↵w
main.c [MySimpleShell] - Code::Blocks 16.01

Debug Tools Plugins Settings Help
main(int argc, const char* argv[]) : int

}

pid_t pid= fork();
int x=0;
if(commandline[1]!= NULL)

{
    x=strcmp(commandline[1],"&"); //if it is a background or foreground process
}

if(pid==0)
{
// if process is child

    if( x == 0 && commandline[1]!= NULL)
    {
        commandline[1]=NULL;
    }

    execvp(inst,commandline);

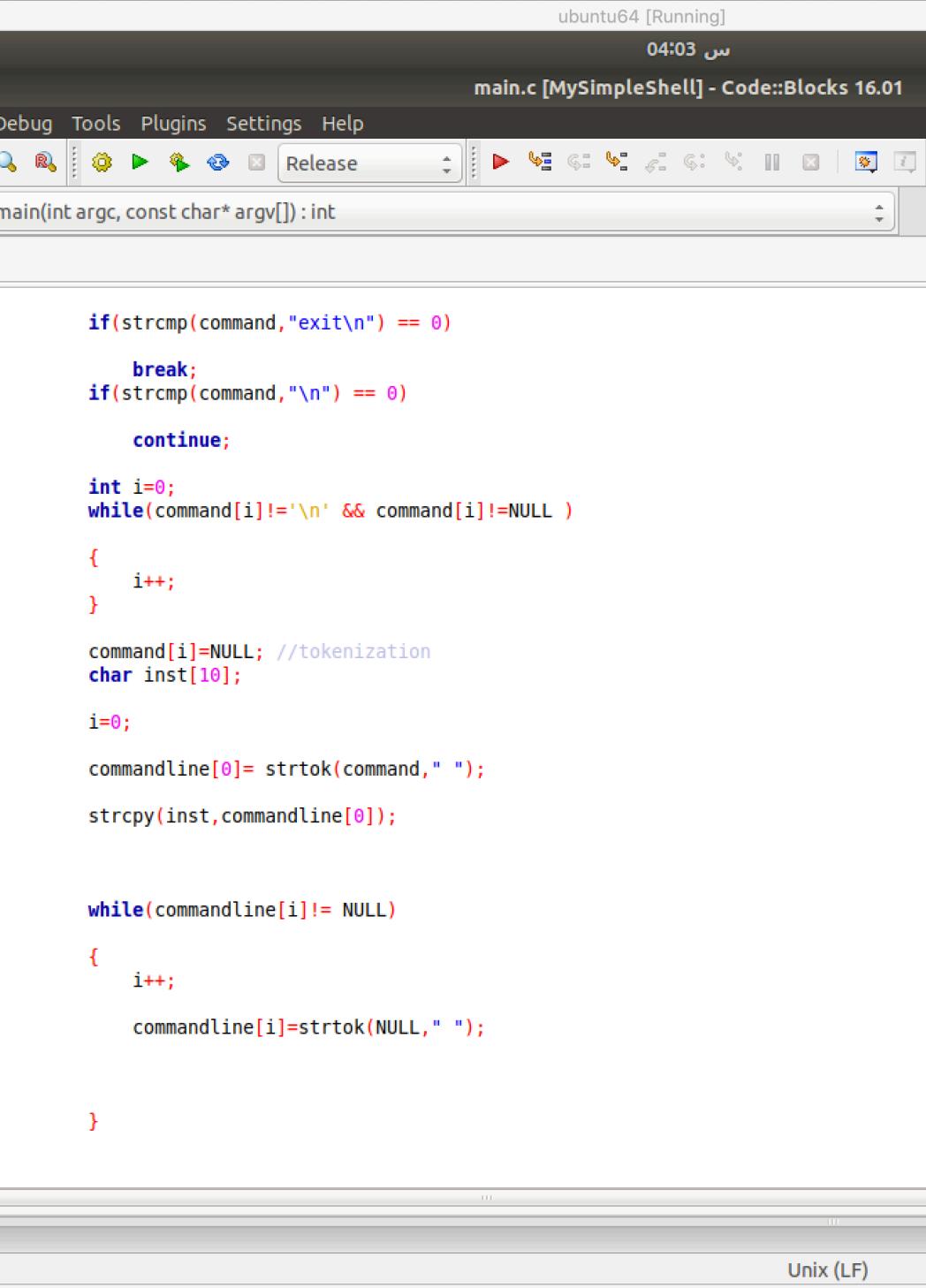
}
else
{// if process is parent
    if( x!=0)
        wait(NULL);
}

}

n.c          Unix (LF)      UTF-8      Line 77, Column 40      Insert
```

The code implements a simple shell functionality. It checks if the second command-line argument is null. If not, it compares it with "&". If it is, it sets the commandline[1] to NULL. Then it uses execvp to execute the command. If the process is a child (pid == 0), it checks if x == 0 and commandline[1] != NULL. If so, it sets commandline[1] to NULL and then executes the command using execvp. If the process is a parent (x != 0), it calls wait(NULL).

This part is responsible for tokenization and handling the exit command and handling blank commands.



The screenshot shows the Code::Blocks IDE interface. The title bar reads "ubuntu64 [Running] 04:03 مـ main.c [MySimpleShell] - Code::Blocks 16.01". The menu bar includes "Debug", "Tools", "Plugins", "Settings", and "Help". Below the menu is a toolbar with various icons. The code editor window displays the following C code:

```
main(int argc, const char* argv[]) : int

if(strcmp(command,"exit\n") == 0)
    break;
if(strcmp(command,"\\n") == 0)
    continue;

int i=0;
while(command[i]!='\n' && command[i]!=NULL )
{
    i++;
}

command[i]=NULL; //tokenization
char inst[10];

i=0;
commandline[0]= strtok(command, " ");
strcpy(inst,commandline[0]);

while(commandline[i]!= NULL)
{
    i++;
    commandline[i]=strtok(NULL, " ");
}

}
```

The status bar at the bottom right shows "Unix (LF)", "UTF-8", "Line 77, Column 40", and "Insert".

createLogFile creates a new text file when the Shell is opened.

signalHandler appends the file every time the process is terminated.

```
L }
void createLogFile ()
{
    FILE *log;
    log=fopen("log", "w");

    fclose(log);

}

signalHandler(int signal)
{
    FILE *log;
    log = fopen("log", "a");

    fprintf(log,"%s","CHild process terminated\n");

    fclose(log);

}

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

```