Problem Statement

Objectives

1. Familiarity with system calls in Unix environment.
2. Introducing processes and their nesting.

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.

Feature

Some of the features present in the simple shell

- The internal shell command "exit" which terminates the shell.
- A command with no arguments.
- A command with arguments.
- A command, with or without arguments, executed in the background using &.
- A pipe separator between various commands.
- Indication of when a process gets forked or terminated.
- Descriptive error and warning messages.

Overall Description

Command parsing subroutines

Tokenizes the line around whitespace and fills the arguments array. It also counts the number of arguments. Termination characters are '|', '&'

```
// token
aI = -1;
bool amp = false;
while (tok) {
     ++aI;
     free(av[aI]);
     av[aI] = (char*) malloc(strlen(tok) + 1);
     strcpy(av[aI], tok);
     if (strcmp(av[aI], "&") == 0)
          amp = true;
     tok = strtok(NULL, " ");
     if (strcmp(av[aI], "|") == 0) {
          --aI;
          break;
     }
}
if (strcmp(av[aI], "&") == 0)
     --aI;
for (int i = aI + 1; i < MAX_ARGS; ++i)
     if (av[i])
          free(av[i]), av[i] = 0;
```

We use char arrays and pointers to handle strings. The function keeps track of whether the command should run in the background.

Command execution subroutines

The method reports process start and termination. For the main process we either wait for the child to finish execution or we leave it for the child signal handler (depending on execution mode background/foreground).

```
     int status;
     pid_t cid = (amp ? vfork() : fork());
     switch (cid) {
     case -1: // can't
          cerr << "can't fork a new process" << endl;
          break;
     case 0: // i'm a child
          if (execvp(av[0], av) == -1) {
               cerr << "no such program \'" << av[0] << "\'" << endl;
               return;
          }
          break;
     default:
          // i'm a parent
          cout << "process[" << cid << "]: started"
                    << (!amp ? " in foreground" : "") << endl;
          if (!amp){
               waitpid(cid, &status, 0);
               cout << "process[" << cid <<
                         "]: terminated with status " << status << endl;
          }
          break;
     }
```

When the child finishes execution, the return signal is catches in the parent process and a message is displayed. Also, the execution of an non-existing programs displays a warning.

Signal handling subroutines

When some event happens at the child process a signal is passed to the parent. So, at first our parent process must register for listening to this signal.

Among these events is the termination of a child process. When a child dies it becomes a zombie and it's up for our handler to wait for its termination and get its ID and return signal. Then, we would get rid of the zombie process and detect its termination.

```
int status = 0;
pid_t p = -1;
do {
     if (p > 0)
          cout << "process[" << p <<
               "]: terminated with status " << status << endl;
     p = waitpid(p, &status, WNOHANG);
} while (p > 0);
signal(SIGCHLD, handle_zombie);
```

Possible Expansion

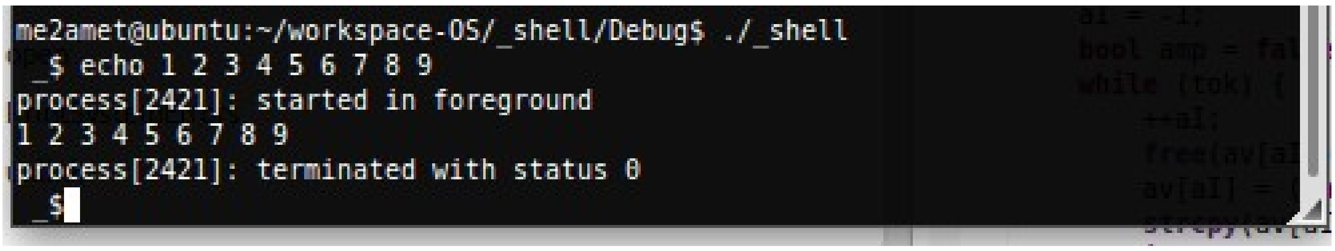
Lots of features can be added to the shell easily

- A default directory can also be implemented.
- Switching between output to stdout and files via arguments.
- Command history.
- Mapping each process ID to the command responsible for its execution.

Sample Runs

Running echo

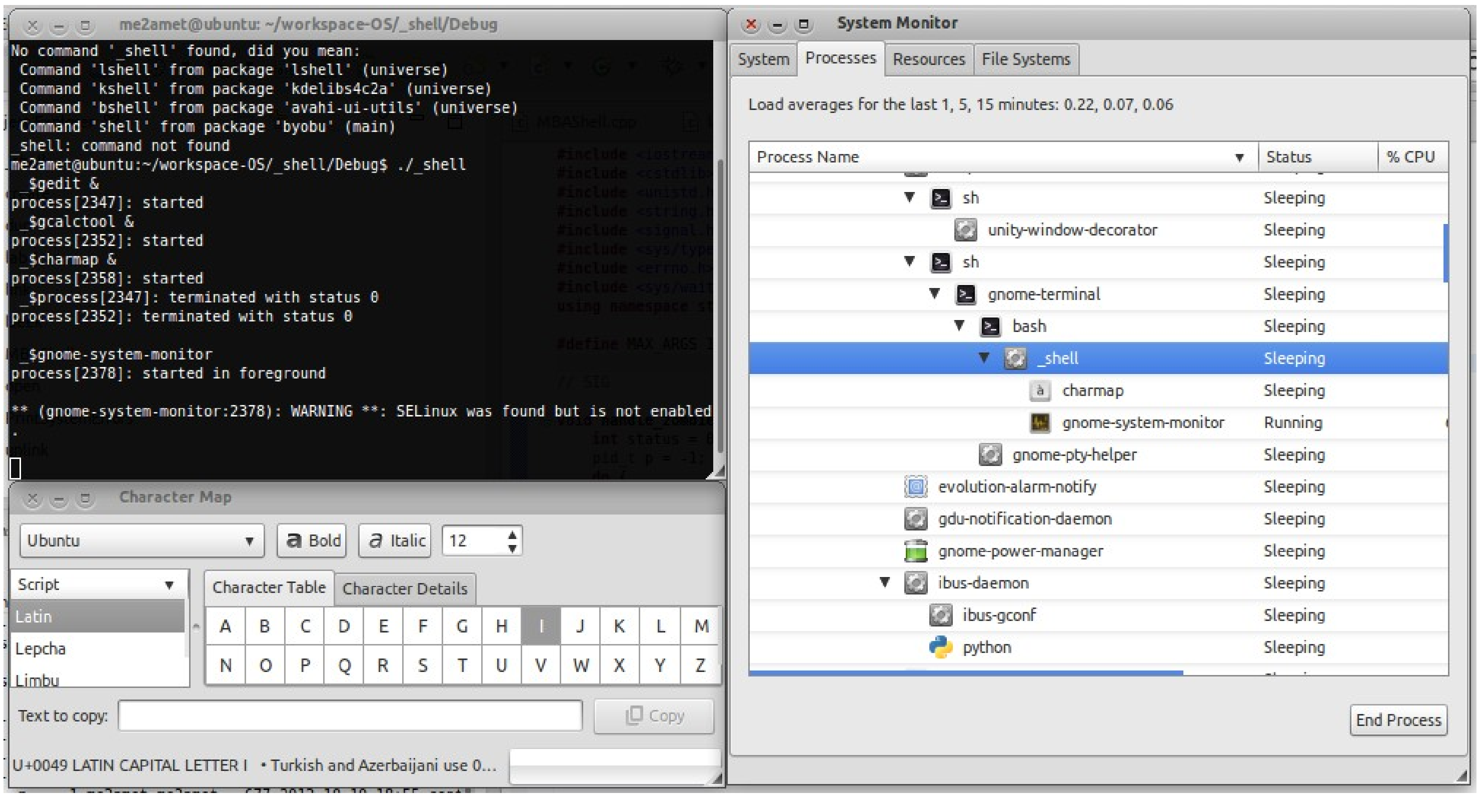
This verifies our argument parsing subroutine.



```
me2amet@ubuntu:~/workspace-OS/_shell/Debug$ ./_shell
 _$ echo 1 2 3 4 5 6 7 8 9
process[2421]: started in foreground
1 2 3 4 5 6 7 8 9
process[2421]: terminated with status 0
 _$
```

Running System Monitor

The process tree is shown and our shell has two child processes.



me2amet@ubuntu: ~/workspace-OS/_shell/Debug

```
No command '_shell' found, did you mean:
 Command 'lshell' from package 'lshell' (universe)
 Command 'kshell' from package 'kdelibs4c2a' (universe)
 Command 'bshell' from package 'avahi-ui-utils' (universe)
 Command 'shell' from package 'byobu' (main)
_shell: command not found
me2amet@ubuntu:~/workspace-OS/_shell/Debug$ ./_shell
 _$gedit &
process[2347]: started
 _$gcalctool &
process[2352]: started
 _$charmap &
process[2358]: started
 _$process[2347]: terminated with status 0
process[2352]: terminated with status 0

 _$gnome-system-monitor
process[2378]: started in foreground

** (gnome-system-monitor:2378): WARNING **: SELinux was found but is not enabled
.
```

Character Map

Ubuntu | Bold | Italic | 12

Script: Latin

Text to copy:

U+0049 LATIN CAPITAL LETTER I • Turkish and Azerbaijani use 0...

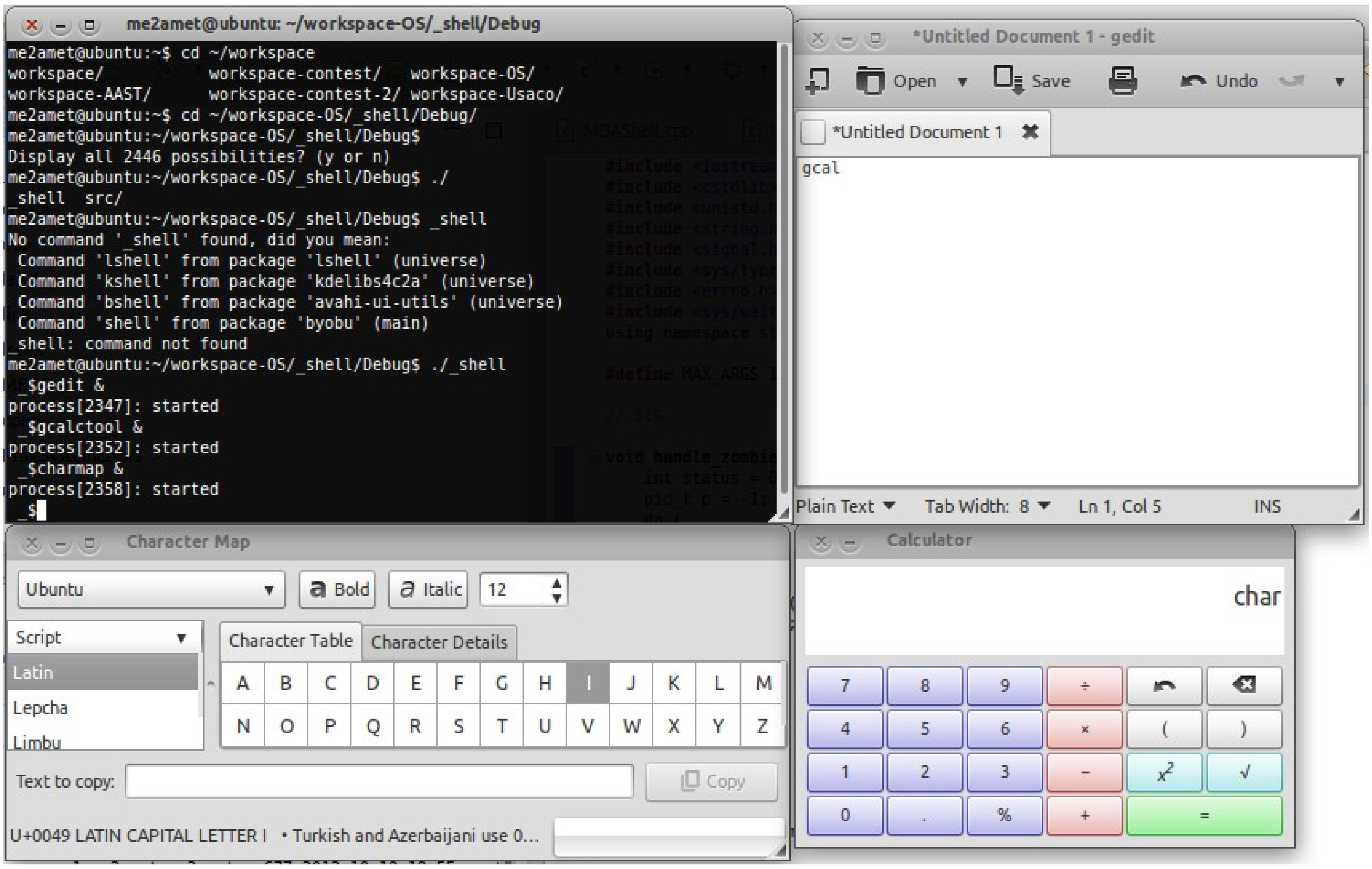
System Monitor

Load averages for the last 1, 5, 15 minutes: 0.22, 0.07, 0.06

Process Name	Status	% CPU
sh	Sleeping	
unity-window-decorator	Sleeping	
sh	Sleeping	
gnome-terminal	Sleeping	
bash	Sleeping	
_shell	Sleeping	
charmap	Sleeping	
gnome-system-monitor	Running	
gnome-pty-helper	Sleeping	
evolution-alarm-notify	Sleeping	
gdu-notification-daemon	Sleeping	
gnome-power-manager	Sleeping	
ibus-daemon	Sleeping	
ibus-gconf	Sleeping	
python	Sleeping	

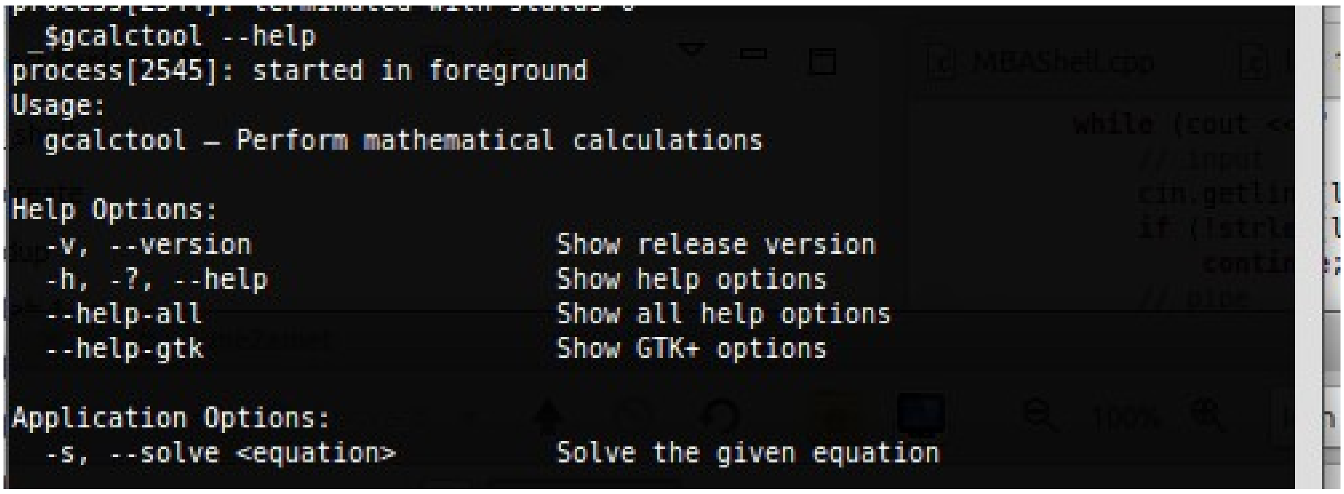
Running three processes in the background

Messages are shown for each process start and its ID.



```
me2amet@ubuntu:~$ cd ~/workspace
workspace/          workspace-contest/   workspace-OS/
workspace-AAST/     workspace-contest-2/ workspace-Usaco/
me2amet@ubuntu:~$ cd ~/workspace-OS/_shell/Debug/
me2amet@ubuntu:~/workspace-OS/_shell/Debug$
Display all 2446 possibilities? (y or n)
me2amet@ubuntu:~/workspace-OS/_shell/Debug$ ./
_shell  src/
me2amet@ubuntu:~/workspace-OS/_shell/Debug$ _shell
No command '_shell' found, did you mean:
 Command 'lshell' from package 'lshell' (universe)
 Command 'kshell' from package 'kdelibs4c2a' (universe)
 Command 'bshell' from package 'avahi-ui-utils' (universe)
 Command 'shell' from package 'byobu' (main)
_shell: command not found
me2amet@ubuntu:~/workspace-OS/_shell/Debug$ ./_shell
 _$gedit &
process[2347]: started
 _$gcalctool &
process[2352]: started
 _$charmap &
process[2358]: started
 _$
```

Running gnome-calc-tool

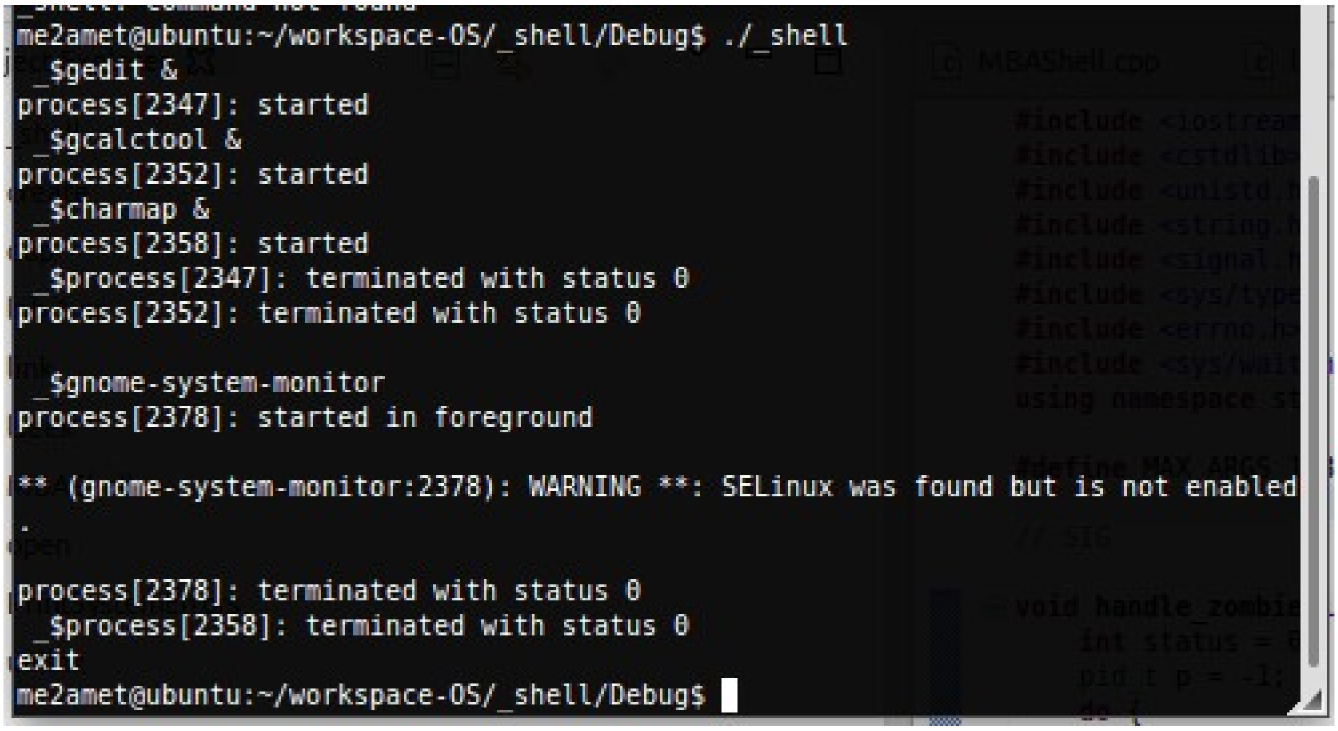


```
 _$gcalctool --help
process[2545]: started in foreground
Usage:
  gcalctool — Perform mathematical calculations

Help Options:
  -v, --version                Show release version
  -h, -?, --help               Show help options
  --help-all                   Show all help options
  --help-gtk                   Show GTK+ options

Application Options:
  -s, --solve <equation>       Solve the given equation
```

Termination signals and exit



```
me2amet@ubuntu:~/workspace-OS/_shell/Debug$ ./_shell
 _$gedit &
process[2347]: started
 _$gcalctool &
process[2352]: started
 _$charmap &
process[2358]: started
 _$process[2347]: terminated with status 0
process[2352]: terminated with status 0

 _$gnome-system-monitor
process[2378]: started in foreground

** (gnome-system-monitor:2378): WARNING **: SELinux was found but is not enabled
.

process[2378]: terminated with status 0
 _$process[2358]: terminated with status 0
exit
me2amet@ubuntu:~/workspace-OS/_shell/Debug$
```