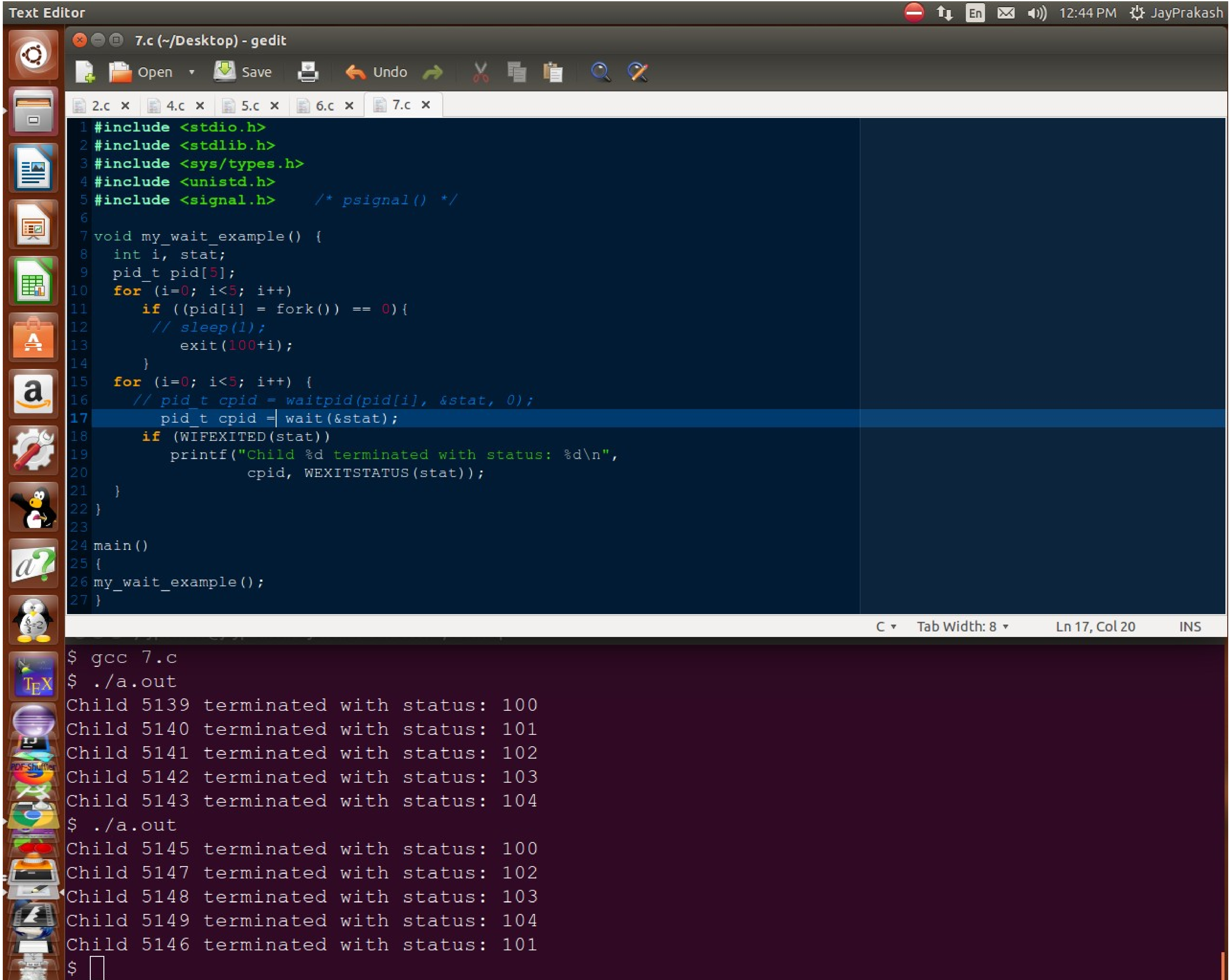
Process Termination

- **If multiple children completed, will reap in arbitrary order**

```
void wait_x() {
  int i, stat;
  pid_t pid[5];
  for (i=0; i<5; i++)
     if ((pid[i] = fork()) == 0){
         sleep(1);
         exit(100+i);
     }
  for (i=0; i<5; i++) {
     pid_t cpid = wait(&stat);
     if (WIFEXITED(stat))
        printf("Child %d terminated with status: %d\n",
                cpid, WEXITSTATUS(stat));
  }
}
```



```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#include <signal.h>    /* psignal() */

void my_wait_example() {
  int i, stat;
  pid_t pid[5];
  for (i=0; i<5; i++)
     if ((pid[i] = fork()) == 0){
      // sleep(1);
        exit(100+i);
     }
  for (i=0; i<5; i++) {
    // pid_t cpid = waitpid(pid[i], &stat, 0);
       pid_t cpid = wait(&stat);
     if (WIFEXITED(stat))
        printf("Child %d terminated with status: %d\n",
                cpid, WEXITSTATUS(stat));
  }
}

main()
{
my_wait_example();
}
```

```
$ gcc 7.c
$ ./a.out
Child 5139 terminated with status: 100
Child 5140 terminated with status: 101
Child 5141 terminated with status: 102
Child 5142 terminated with status: 103
Child 5143 terminated with status: 104
$ ./a.out
Child 5145 terminated with status: 100
Child 5147 terminated with status: 102
Child 5148 terminated with status: 103
Child 5149 terminated with status: 104
Child 5146 terminated with status: 101
$
```

waitpid(): waiting for a specific process

- Useful when parent has more than one child, or you want to check for exited child but not block

The child to wait for/check on
-1 means any child

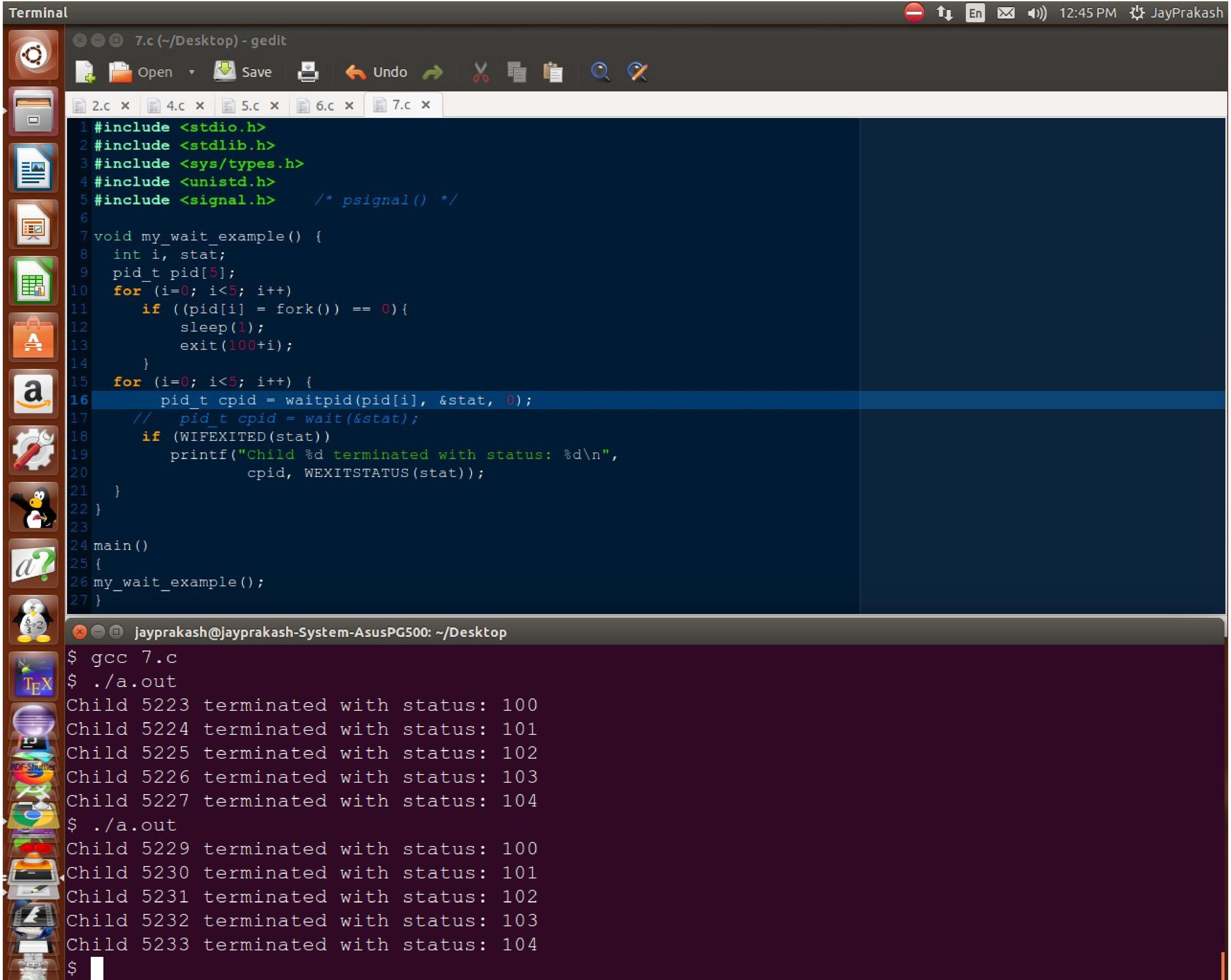
```
pid_t  result =
     waitpid(child_pid,
             &status,
             options);
```

0 = no options, wait until child exits
WNOHANG = don't wait, just check

- Return value
  - pid of child, if child has exited
  - 0, if using WNOHANG and child hasn't exited

- **Can use waitpid() to reap in order**

```
void wait_x() {
  int i, stat;
  pid_t pid[5];
  for (i=0; i<5; i++)
     if ((pid[i] = fork()) == 0){
         sleep(1);
         exit(100+i);
     }
  for (i=0; i<5; i++) {
     pid_t cpid = waitpid(pid[i], &stat, 0);
     if (WIFEXITED(stat))
        printf("Child %d terminated with status: %d\n",
                cpid, WEXITSTATUS(stat));
  }
}
```



Terminal

7.c (~/Desktop) - gedit

2.c 4.c 5.c 6.c 7.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#include <signal.h>    /* psignal() */

void my_wait_example() {
  int i, stat;
  pid_t pid[5];
  for (i=0; i<5; i++)
     if ((pid[i] = fork()) == 0){
         sleep(1);
         exit(100+i);
     }
  for (i=0; i<5; i++) {
       pid_t cpid = waitpid(pid[i], &stat, 0);
    //   pid_t cpid = wait(&stat);
     if (WIFEXITED(stat))
        printf("Child %d terminated with status: %d\n",
                cpid, WEXITSTATUS(stat));
  }
}

main()
{
my_wait_example();
}
```

jayprakash@jayprakash-System-AsusPG500: ~/Desktop

```
$ gcc 7.c
$ ./a.out
Child 5223 terminated with status: 100
Child 5224 terminated with status: 101
Child 5225 terminated with status: 102
Child 5226 terminated with status: 103
Child 5227 terminated with status: 104
$ ./a.out
Child 5229 terminated with status: 100
Child 5230 terminated with status: 101
Child 5231 terminated with status: 102
Child 5232 terminated with status: 103
Child 5233 terminated with status: 104
$
```