

Reaping Children

- **wait(): parents reap their dead children**
 - Given info about why child died, exit status, etc.
- **Two variants**
 - wait(): wait for and reap next child to exit
 - waitpid(): wait for and reap specific child

```
pid_t wait(int *stat_loc);
```

when called by a process with ≥ 1 children:

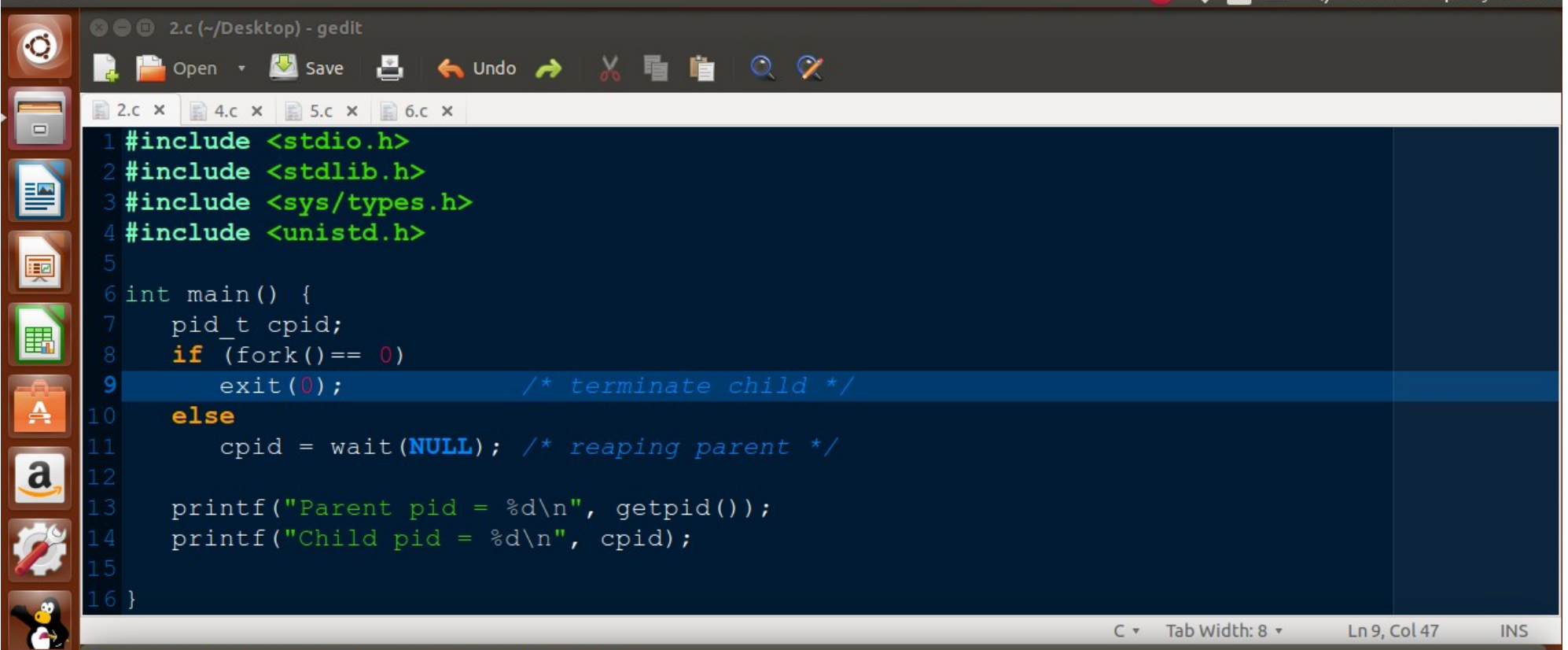
- *waits* (if needed) for a child to terminate
- *reaps* a terminated child (if ≥ 1 terminated children, arbitrarily pick one)
- *returns* reaped child's pid and exit status info via pointer (if non-NULL)

when called by a process with no children:

- return -1 immediately

Example program

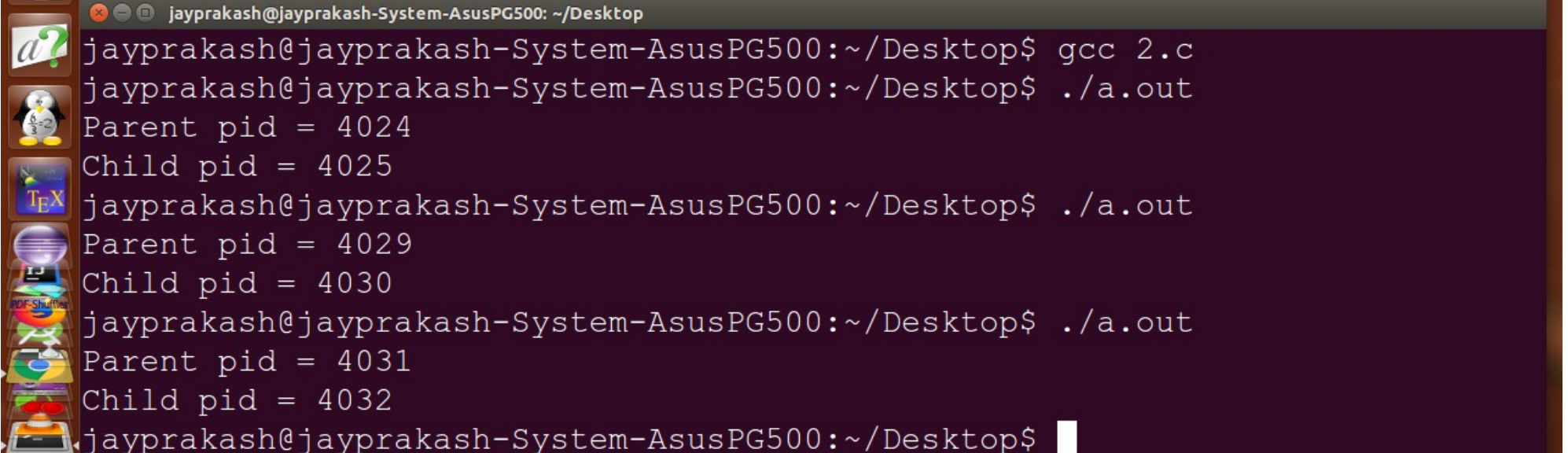
```
int main() {  
    pid_t cpid;  
    if (fork() == 0)  
        exit(0);           /* terminate child */  
    else  
        cpid = wait(NULL); /* reaping parent */  
  
    printf("Parent pid = %d\n", getpid());  
    printf("Child pid = %d\n", cpid);  
  
}
```



The image shows a Gedit text editor window titled "2.c (~/Desktop) - gedit". The editor has a menu bar with "Open", "Save", "Undo", and other standard options. Below the menu bar are tabs for "2.c", "4.c", "5.c", and "6.c". The main editing area shows a C program with the following code:

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <sys/types.h>
4 #include <unistd.h>
5
6 int main() {
7     pid_t cpid;
8     if (fork() == 0)
9         exit(0);          /* terminate child */
10    else
11        cpid = wait(NULL); /* reaping parent */
12
13    printf("Parent pid = %d\n", getpid());
14    printf("Child pid = %d\n", cpid);
15
16 }
```

The status bar at the bottom of the editor shows "C", "Tab Width: 8", "Ln 9, Col 47", and "INS".



The image shows a terminal window titled "jayprakash@jayprakash-System-AsusPG500: ~/Desktop". The terminal displays the following commands and output:

```
jayprakash@jayprakash-System-AsusPG500:~/Desktop$ gcc 2.c
jayprakash@jayprakash-System-AsusPG500:~/Desktop$ ./a.out
Parent pid = 4024
Child pid = 4025
jayprakash@jayprakash-System-AsusPG500:~/Desktop$ ./a.out
Parent pid = 4029
Child pid = 4030
jayprakash@jayprakash-System-AsusPG500:~/Desktop$ ./a.out
Parent pid = 4031
Child pid = 4032
jayprakash@jayprakash-System-AsusPG500:~/Desktop$
```

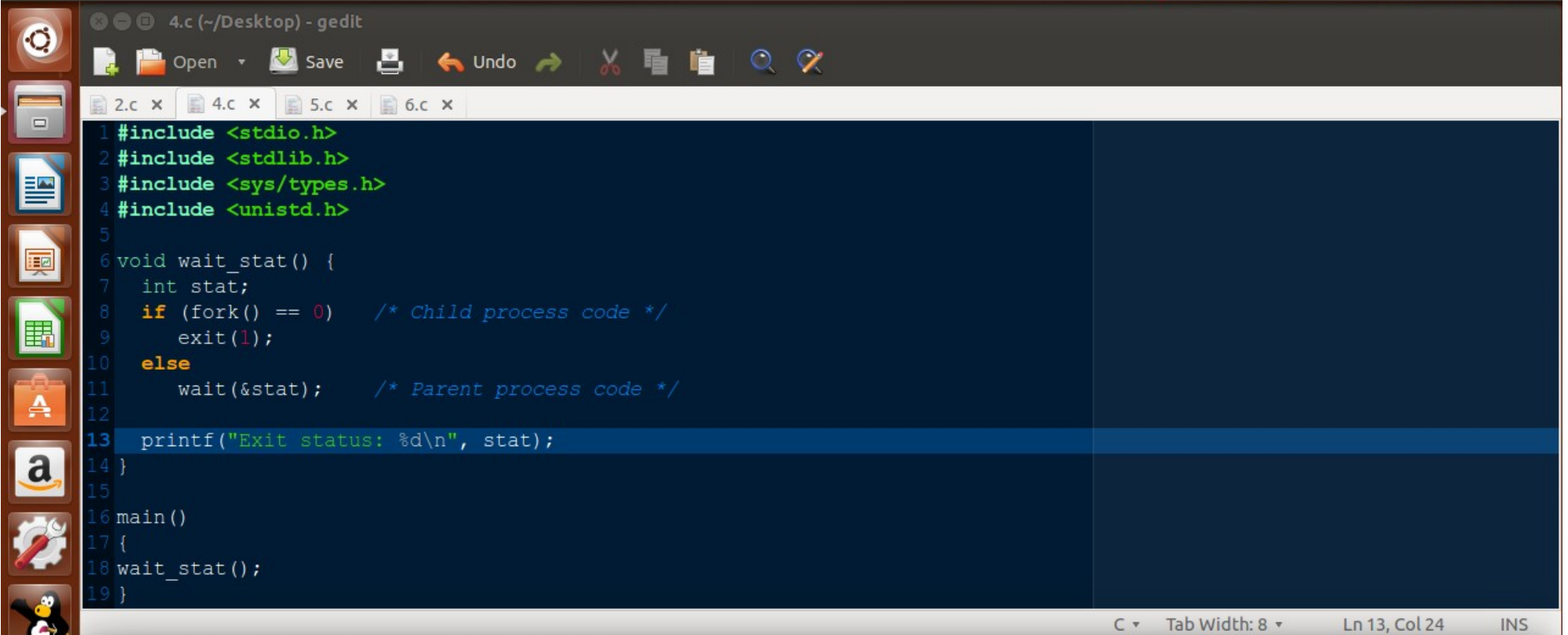
Example program

```
int main() {  
    if (fork() == 0) {  
        printf("HC: hello from child\n");  
    } else {  
        printf("HP: hello from parent\n");  
        wait(NULL);  
        printf("CT: child has terminated\n");  
    }  
    printf("Bye\n");  
}
```

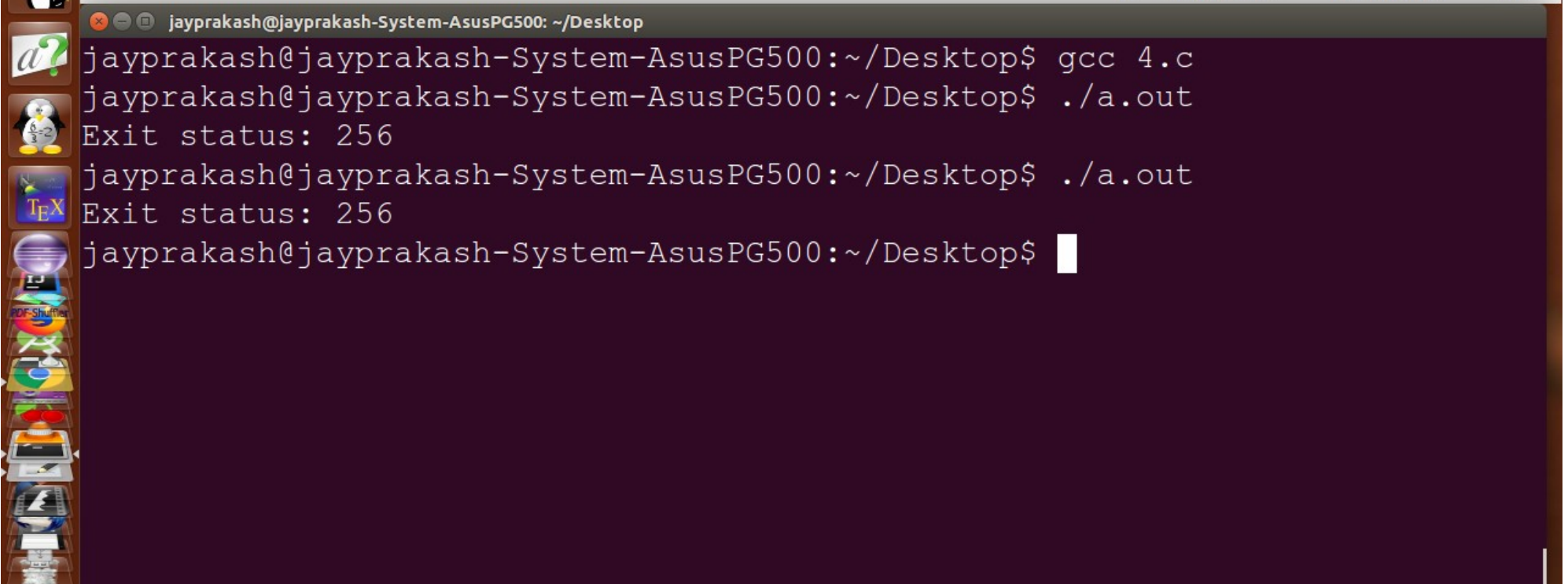


Child status information

- **status information about the child reported by wait is more than just the exit status of the child**
 - normal/abnormal termination
 - termination cause
 - exit status



```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <sys/types.h>
4 #include <unistd.h>
5
6 void wait_stat() {
7     int stat;
8     if (fork() == 0) /* Child process code */
9         exit(1);
10    else
11        wait(&stat); /* Parent process code */
12
13    printf("Exit status: %d\n", stat);
14 }
15
16 main()
17 {
18     wait_stat();
19 }
```



```
jayprakash@jayprakash-System-AsusPG500: ~/Desktop
jayprakash@jayprakash-System-AsusPG500:~/Desktop$ gcc 4.c
jayprakash@jayprakash-System-AsusPG500:~/Desktop$ ./a.out
Exit status: 256
jayprakash@jayprakash-System-AsusPG500:~/Desktop$ ./a.out
Exit status: 256
jayprakash@jayprakash-System-AsusPG500:~/Desktop$
```

Additional Status Info from wait() system Call

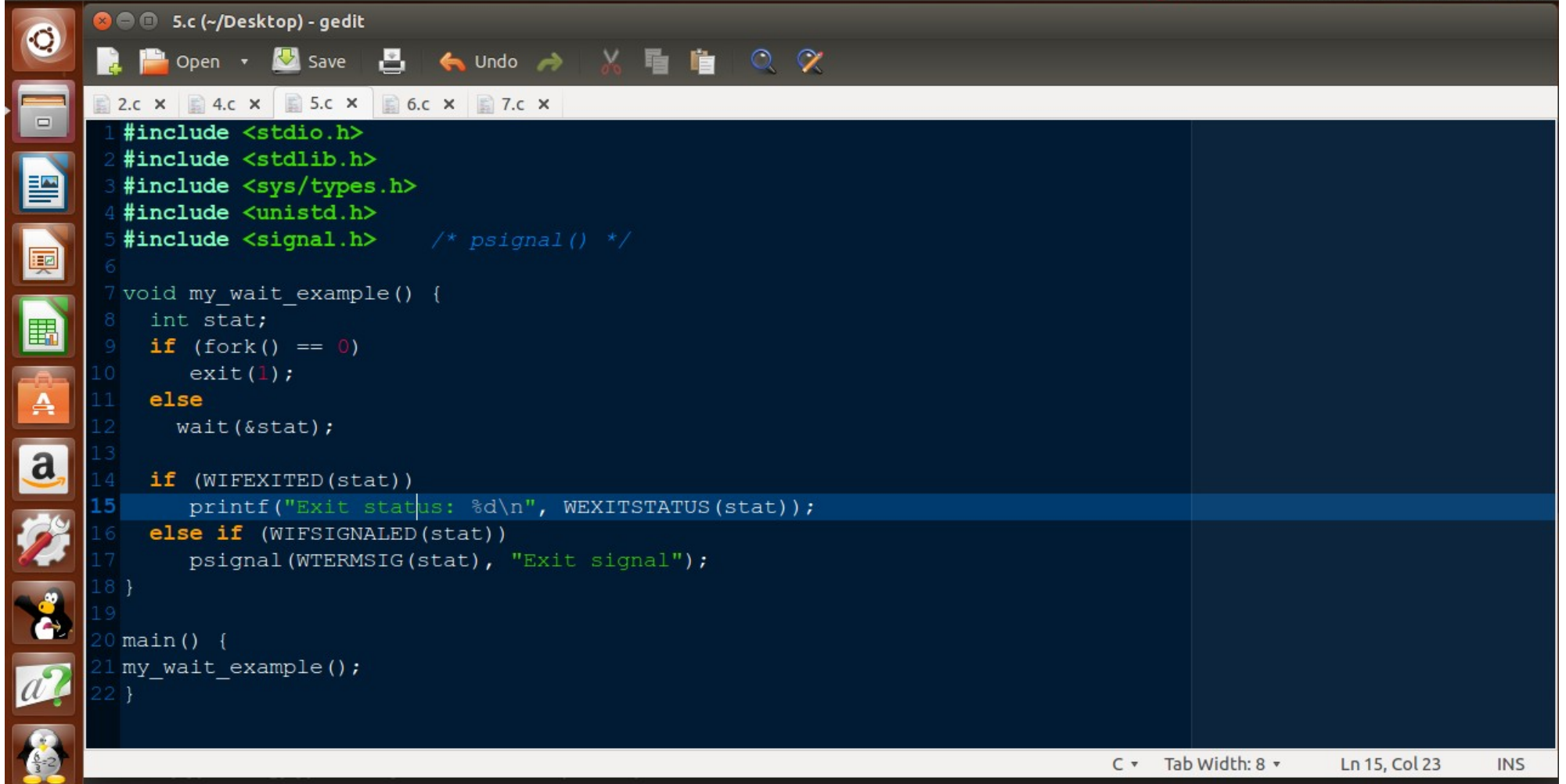
childpid = wait(&wstatus); → **returns the exit status from child which can further be inspected using the following macros.**

The WIF ... Macros

- **WIFEXITED(wstatus)** → **returns true** if child terminated normally.
 - WEXITSTATUS(wstatus) → returns exit status (least significant 8 bits)
- **WIFSIGNALED(wstatus)** → **returns true** if child process was terminated by a signal.
 - WTERMSIG(wstatus) → returns the number of signal
- **WCOREDUMP(wstatus)** → **returns true** if child produced a core dump
- **WIFSTOPPED(wstatus)** → **returns true** if child was stopped by a signal
 - WSTOPSIG(wstatus) → returns the signal number which caused child to stop
- **WIFCONTINUED(wstatus)** → **returns true** if child was resumed with SIGCONT signal

```
/* prints information about a signal */
```

```
void psignal(unsigned sig, const char *s);
```

The image shows a screenshot of a Linux desktop environment. On the left side, there is a vertical dock containing several application icons, including the Dash icon, Home icon, Files icon, LibreOffice Writer, LibreOffice Calc, LibreOffice Impress, Amazon, System Settings, a web browser, a terminal, and a few others. The main window is a text editor titled "5.c (~/Desktop) - gedit". It contains a C program that demonstrates the use of the `psignal()` function. The code is as follows:

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <sys/types.h>
4 #include <unistd.h>
5 #include <signal.h>    /* psignal() */
6
7 void my_wait_example() {
8     int stat;
9     if (fork() == 0)
10         exit(1);
11     else
12         wait(&stat);
13
14     if (WIFEXITED(stat))
15         printf("Exit status: %d\n", WEXITSTATUS(stat));
16     else if (WIFSIGNALED(stat))
17         psignal(WTERMSIG(stat), "Exit signal");
18 }
19
20 main() {
21     my_wait_example();
22 }
```

The status bar at the bottom of the text editor shows "C", "Tab Width: 8", "Ln 15, Col 23", and "INS".

```
$ gcc 5.c
$ ./a.out
Exit status: 1
$ ./a.out
Exit status: 1
$
```

6.c (~/Desktop) - gedit

Open Save Undo

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

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <sys/types.h>
4 #include <unistd.h>
5 #include <signal.h> /* psignal() */
6
7 void my_wait_example() {
8     int stat;
9     if (fork() == 0){
10         sleep(30); exit(1);
11     }else
12         wait(&stat);
13
14     if (WIFEXITED(stat))
15         printf("Exit status: %d\n", WEXITSTATUS(stat));
16     else if (WIFSIGNALED(stat))
17         psignal(WTERMSIG(stat), "Exit signal");
18 }
19
20 main() {
21     my_wait_example();
22 }
```

jayprakash@jayprakash-System-AsusPG500: ~

```
$ ps -a
  PID TTY          TIME CMD
 4847 pts/0        00:00:00 6.o
 4848 pts/0        00:00:00 6.o
 4849 pts/9        00:00:00 ps
$ kill -9 4848
$
```

jayprakash@jayprakash-System-AsusPG500: ~/Desktop

```
$ gcc 6.c -o 6.o
```

```
$ ./6.o
```

```
Exit signal: Killed
```

```
$
```

6.c (~/Desktop) - gedit

Open Save Undo

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

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <sys/types.h>
4 #include <unistd.h>
5 #include <signal.h> /* psignal() */
6
7 void my_wait_example() {
8     int stat;
9     if (fork() == 0){
10         sleep(30); exit(1);
11     }else
12         wait(&stat);
13
14     if (WIFEXITED(stat))
15         printf("Exit status: %d\n", WEXITSTATUS(stat));
16     else if (WIFSIGNALED(stat))
17         psignal(WTERMSIG(stat), "Exit signal");
18 }
19
20 main() {
21     my_wait_example();
22 }
```

jayprakash@jayprakash-System-AsusPG500: ~

```
$ ps -a
  PID TTY          TIME CMD
 4894 pts/0        00:00:00 6.o
 4895 pts/0        00:00:00 6.o
 4896 pts/9        00:00:00 ps
$ kill -15 4895
$
```

jayprakash@jayprakash-System-AsusPG500: ~/Desktop

```
$ gcc 6.c -o 6.o
```

```
$ ./6.o
```

```
Exit signal: Terminated
```

```
$
```


6.c (~/Desktop/SS 2021/Wk-4/example code/wait) - gedit

Open Save Undo

6.c x

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <sys/types.h>
4 #include <unistd.h>
5 #include <signal.h> /* psignal() */
6
7 void my_wait_example() {
8     int stat;
9     if (fork() == 0){
10         sleep(30); exit(1);
11     }else
12         wait(&stat);
13
14     if (WIFEXITED(stat))
15         printf("Exit status: %d\n", WEXITSTATUS(stat));
16     else if (WIFSIGNALED(stat))
17         psignal(WTERMSIG(stat), "Exit signal");
18     /*kill -15(Terminate), -9(Kill), -3(Quit) *
19     /* -1(Hangup parent) - psignal not executed in parent as it is hanged up */
20     /* Check the output of command kill -l for a list of possible signals */
21 }
22 main() {
23 my_wait_example();
24 }
```

\$ gcc 6.c -o 6.o

\$./6.o

Exit signal: Quit

\$./6.o

Exit signal: Terminated

\$./6.o

Hangup

\$ kill -l

1) SIGHUP	2) SIGINT	3) SIGQUIT	4) SIGILL	5) SIGTRAP
6) SIGABRT	7) SIGBUS	8) SIGFPE	9) SIGKILL	10) SIGUSR1
11) SIGSEGV	12) SIGUSR2	13) SIGPIPE	14) SIGALRM	15) SIGTERM
16) SIGSTKFLT	17) SIGCHLD	18) SIGCONT	19) SIGSTOP	20) SIGTSTP
21) SIGTTIN	22) SIGTTOU	23) SIGURG	24) SIGXCPU	25) SIGXFSZ
26) SIGVTALRM	27) SIGPROF	28) SIGWINCH	29) SIGIO	30) SIGPWR

jayprakash@jayprakash-System-AsusPG500:

```
$ ps -a
  PID TTY          TIME CMD
 8894 pts/0        00:00:00 6.o
 8895 pts/0        00:00:00 6.o
 8896 pts/10      00:00:00 ps
$ kill -3 8895
$ ps -a
  PID TTY          TIME CMD
 8898 pts/0        00:00:00 6.o
 8899 pts/0        00:00:00 6.o
 8902 pts/10      00:00:00 ps
$ kill -15 8899
$ ps -a
  PID TTY          TIME CMD
 8983 pts/0        00:00:00 6.o
 8984 pts/0        00:00:00 6.o
 8988 pts/10      00:00:00 ps
$ kill -1 8983
$
```

C Tab Width: 8 Ln 23, Col 19 INS

6.c (~/Desktop/SS 2021/Wk-4/example code/wait) - gedit

Open Save Undo

6.c x

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <sys/types.h>
4 #include <unistd.h>
5 #include <signal.h> /* psignal() */
6
7 void my_wait_example() {
8     int stat;
9     if (fork() == 0){ int i=1/0; // generates signal at run-time
10        sleep(30); exit(1);
11    }else
12        wait(&stat);
13
14    if (WIFEXITED(stat))
15        printf("Exit status: %d\n", WEXITSTATUS(stat));
16    else if (WIFSIGNALED(stat))
17        psignal(WTERMSIG(stat), "Exit signal");
18    /*kill -15(Terminate), -9(Kill), -3(Quit) *
19    /* -1(Hangup parent) - psignal not executed in parent as it is hanged up */
20    /* Check the output of command kill -l for a list of possible signals */
21 }
22 main() {
23     my_wait_example();
24 }
```

jayprakash@jayprakash-System-AsusPG500: ~/Desktop/SS 2021/Wk-4/example code/wait

\$ gcc 6.c -o 6.o

6.c: In function 'my_wait_example':

6.c:9:28: warning: division by zero [-Wdiv-by-zero]

if (fork() == 0){ int i=1/0; // generates signal at run-time

^

\$./6.o

Exit signal: Floating point exception

\$