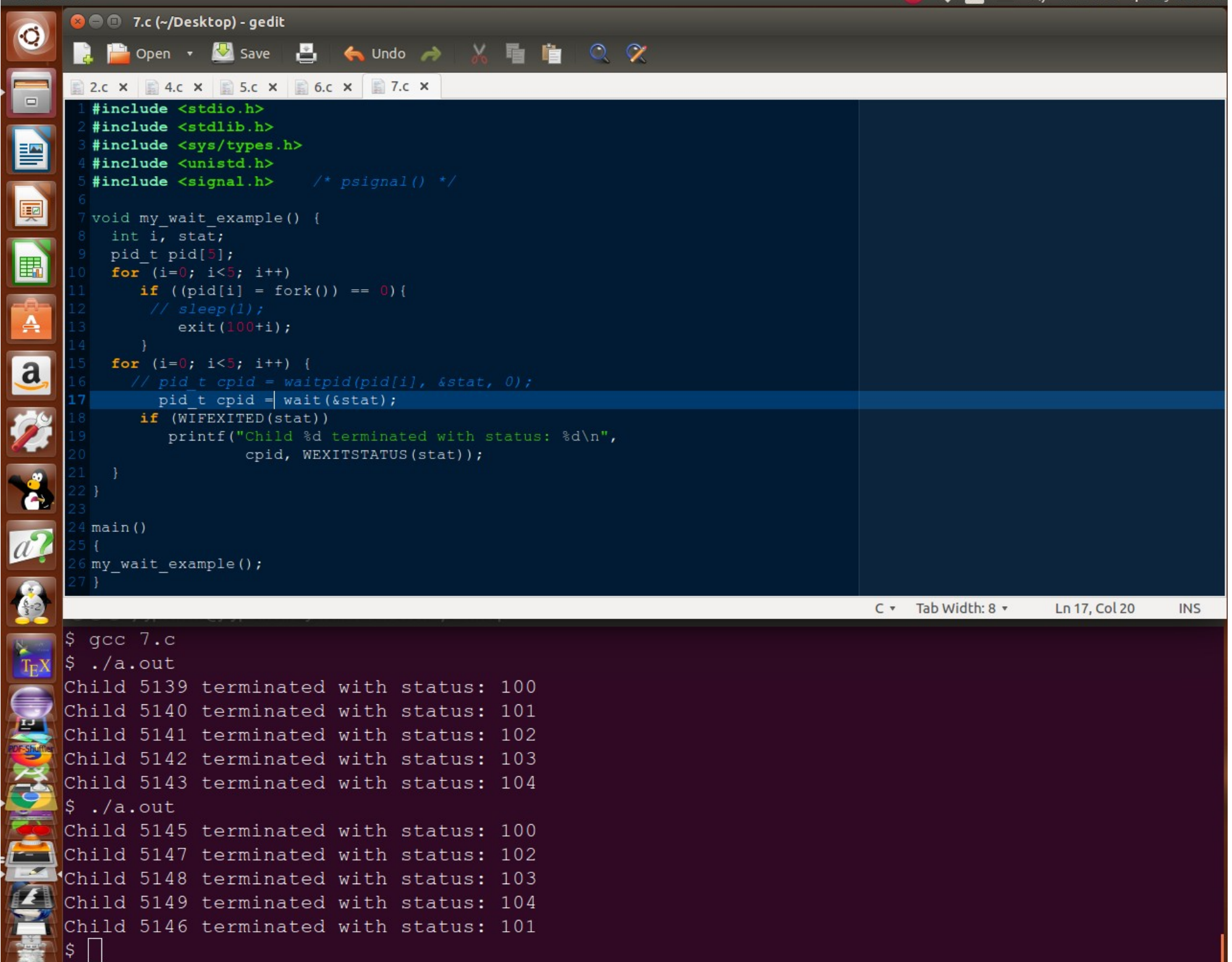


# Process Termination

- 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));
    }
}
```



The screenshot shows a Linux desktop environment. On the left is a vertical dock with various application icons including a gear, a folder, a document, a spreadsheet, a presentation, an Amazon logo, a wrench and screwdriver, a penguin, a question mark, another penguin, and a terminal icon. The main window is a text editor titled "7.c (~/Desktop) - gedit". It contains a C program that forks five child processes, each sleeping for a different duration (1 to 5 seconds) before exiting with a status value from 100 to 104. 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 i, stat;
9     pid_t pid[5];
10    for (i=0; i<5; i++)
11        if ((pid[i] = fork()) == 0){
12            // sleep(1);
13            exit(100+i);
14        }
15    for (i=0; i<5; i++) {
16        // pid_t cpid = waitpid(pid[i], &stat, 0);
17        pid_t cpid = wait(&stat);
18        if (WIFEXITED(stat))
19            printf("Child %d terminated with status: %d\n",
20                  cpid, WEXITSTATUS(stat));
21    }
22 }
23
24 main()
25 {
26     my_wait_example();
27 }
```

Below the text editor is a terminal window showing the compilation and execution of the program:

```
$ 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
$
```

The status bar at the bottom of the text editor indicates "C", "Tab Width: 8", "Ln 17, Col 20", and "INS".

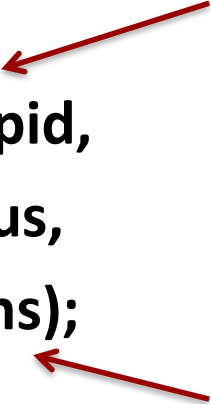
# waitpid(): waiting for a specific process

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

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

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

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

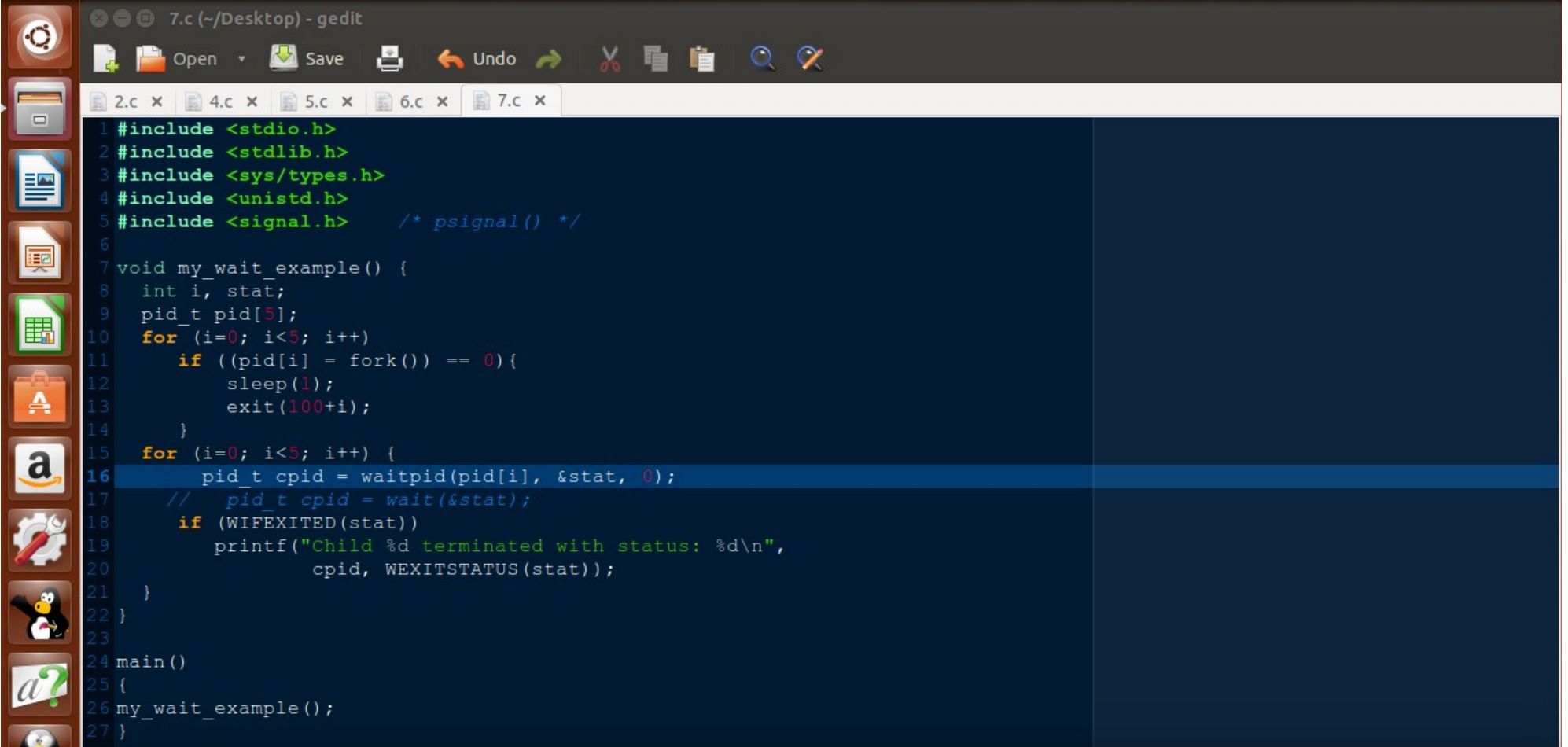


## ■ Return value

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

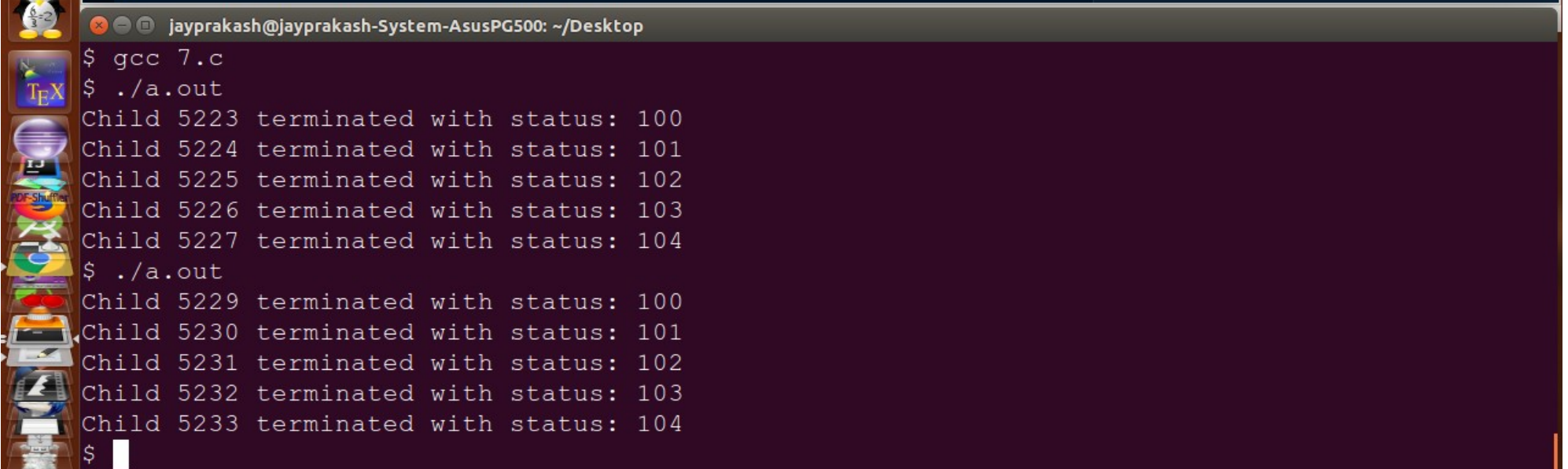
## ■ Can use waitpid() to reap in order

```
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));
    }
}
```



The image shows a Gedit editor window titled "7.c (~/Desktop) - gedit". The editor contains a C program that demonstrates process synchronization using `waitpid()`. The program includes headers for `stdio.h`, `stdlib.h`, `sys/types.h`, `unistd.h`, and `signal.h`. It defines a function `my_wait_example()` that forks five child processes. Each child process sleeps for a duration equal to its PID (100+i) and then exits with a status of 100+i. The parent process then waits for each child using `waitpid()` and prints a message indicating the child's termination status.

```
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 i, stat;
9     pid_t pid[5];
10    for (i=0; i<5; i++)
11        if ((pid[i] = fork()) == 0){
12            sleep(1);
13            exit(100+i);
14        }
15    for (i=0; i<5; i++) {
16        pid_t cpid = waitpid(pid[i], &stat, 0);
17        // pid_t cpid = wait(&stat);
18        if (WIFEXITED(stat))
19            printf("Child %d terminated with status: %d\n",
20                cpid, WEXITSTATUS(stat));
21    }
22 }
23
24 main()
25 {
26    my_wait_example();
27 }
```



The image shows a terminal window titled "jayprakash@jayprakash-System-AsusPG500: ~/Desktop". It displays the compilation and execution of the C program. The program is compiled with `gcc 7.c` and executed with `./a.out`. The output shows five child processes terminating with status 100 through 104, followed by another execution of `./a.out` showing five more child processes terminating with status 100 through 104.

```
$ 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
$
```