四川大學
SICHUAN UNIVERSITY

The Experiment Of OS（3）

Multiple-Process

Xinsheng Li 李新胜

10/28/2024

Nothing in the world is difficult for one who sets his mind to it.
世上无难事, 只怕有心人。

Self-introduction

- Name: Xinsheng Li 李新胜
- E-mail: lixinsheng@scu.edu.cn
- My field: Digital image processing, Computer vision, Simulation training.

Content

- Installation of virtual machine Vmware, Ubuntu 16.04, Vmware tools
- The basic commands for Linux
- VI, Shell，Gcc，Gdb and CMake
- **Multi-process**
- Multi-thread

Overview

- Gcc, Gdb
- Command: ps
- Multiple-Process, Fork
- Advices:
  - Program with patience
  - Problems improve you gradually. Search the solutions online.
  - Try smaller examples when you do not understand.
  - Use IDE if you can. VSCode and cmake are recommended.
  - Debug your code on windows. Then run it on Linux.

GCC

GCC is a powerful tool set, it contains preprocessor, compiler, assembler, linker and other components. It will call the appropriate components according to the optioned passed to GCC.

GCC command formation：

gcc [options] infile…. –o outpile

Install GCC

sudo apt update

sudo apt install build-essential

GCC Compiler(cont.)

Steps for GCC Compile Process

There are four steps for GCC Compile Process:

（1）Pre-Processing: -E

In this phrase, the preprocessor directives will be expand. The postfix of output file is .i.

（2）Compiling: -S

In this phrase, GCC will checks whether there is syntax error in the program. If there is no error in the program, Gcc will translate the c source code to assemble format without creating an object file. The postfix of output file is .s.

GCC Compile(cont.)

（3）Assembling: -c

In this phrase, Gcc will translate the assemble code into machine instruction, and generate object file. The postfix of output file is .o.

（4）Linking: -o

The final phrase is Linking ,in this phrase, Gcc will link object files to produce final executable file.

GCC Compiler(cont.)

GCC Common Options

GCC command formation：gcc [options] infile…. –o outpile

-c compile and generate object file, but do not connect links other objects file

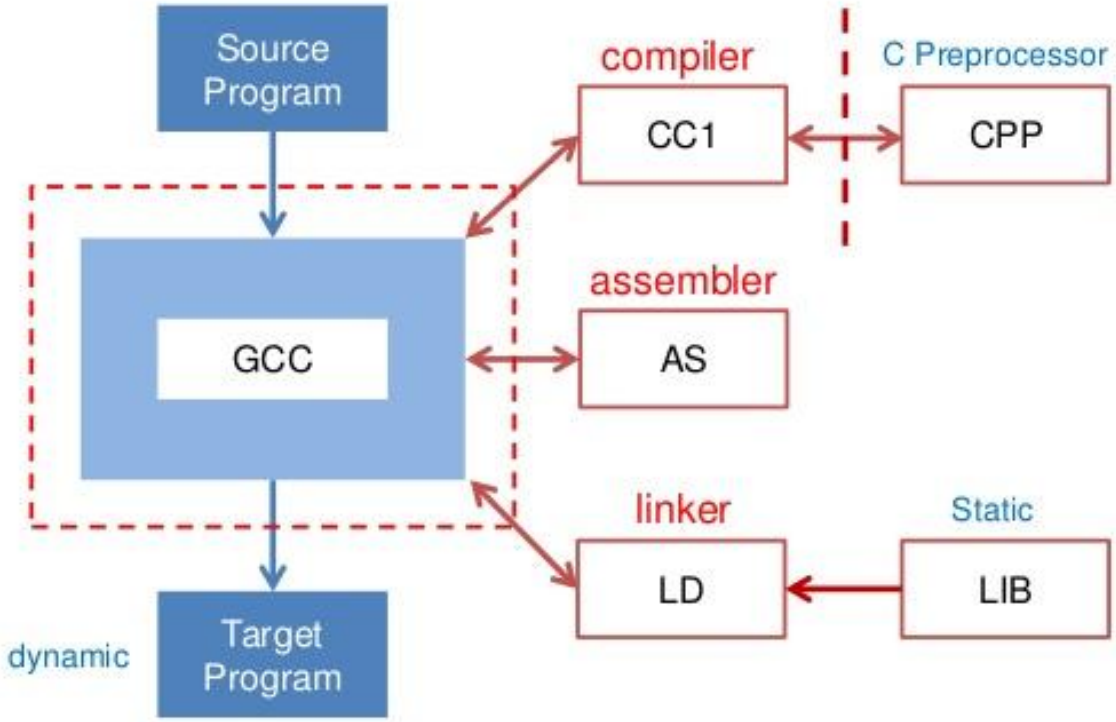
-S covert the source file to assemble file

-E expand the processor directives of the source code

-g generate the debug information which can be used by gdb

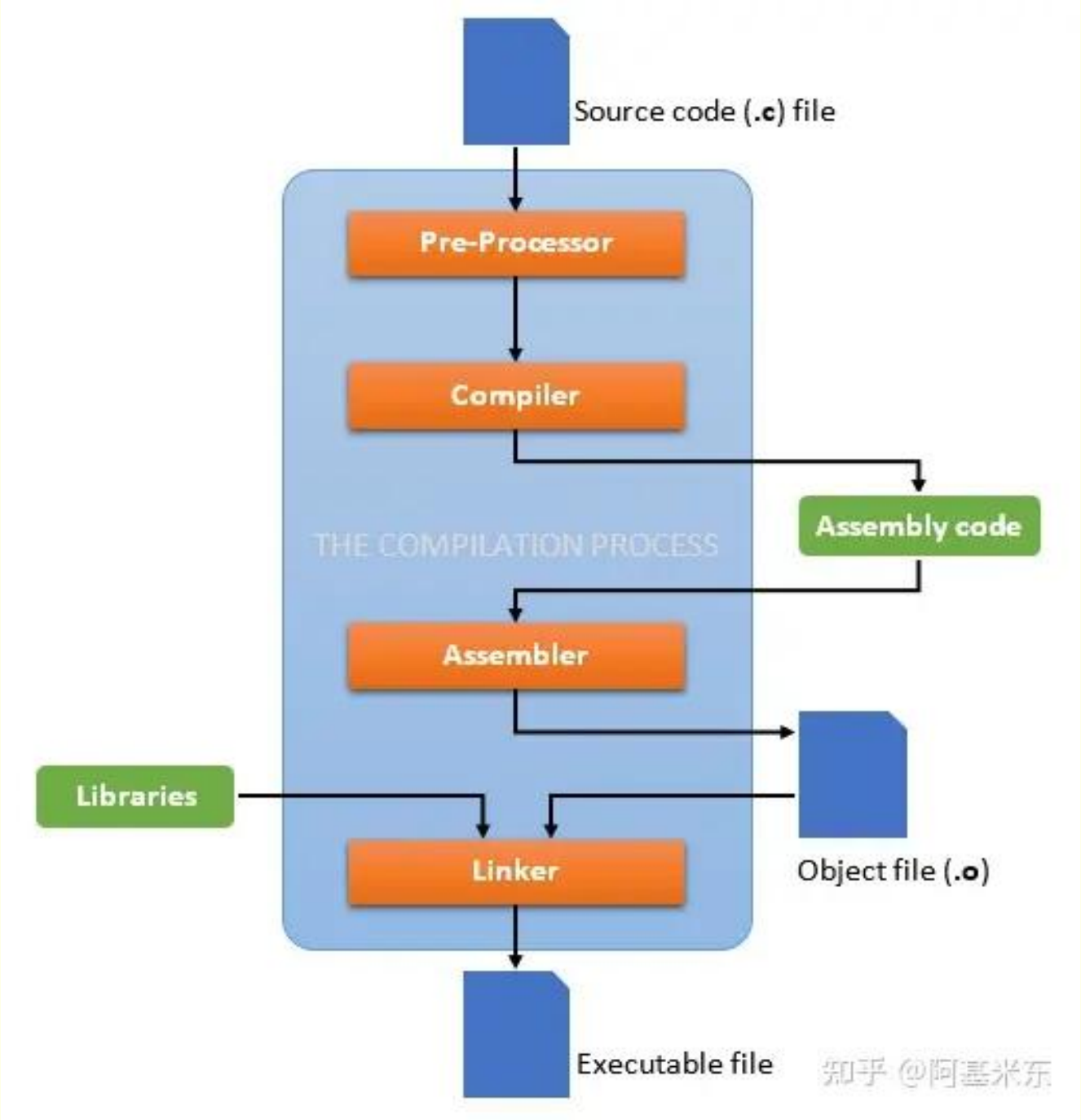
-o output filename: specifies the filename for the output file, if you do not use this option, the filename for the output file is a.out.

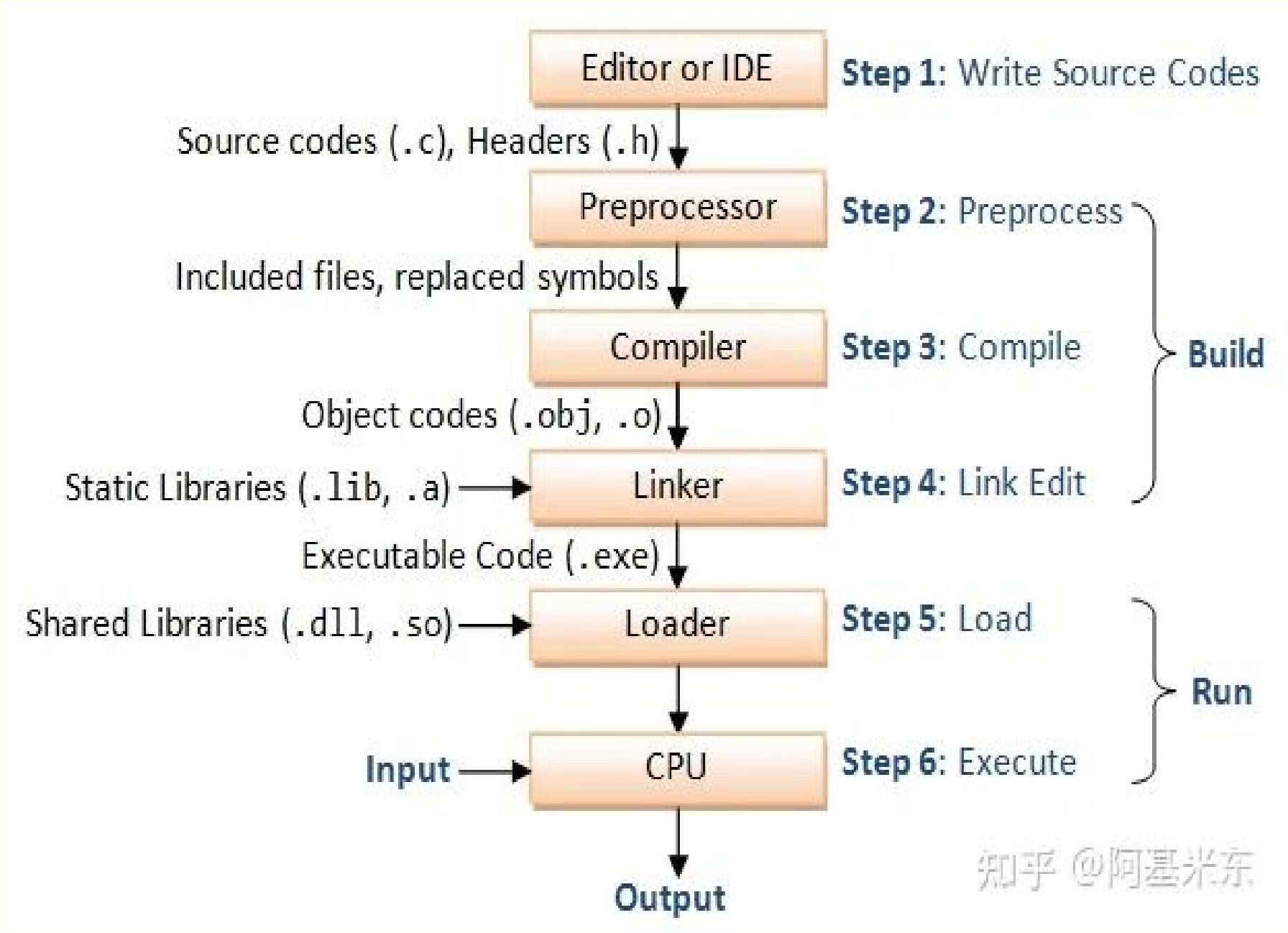
GCC compiler



GCC is a collection that invokes compiler, assembler and linker…

13


Source code (.c) file
Pre-Processor
Compiler
Assembly code
THE COMPILATION PROCESS
Assembler
Libraries
Linker
Object file (.o)
Executable file
知乎 @阿基米东


Editor or IDE
Step 1: Write Source Codes
Source codes (.c), Headers (.h)
Preprocessor
Step 2: Preprocess
Included files, replaced symbols
Compiler
Step 3: Compile
Build
Object codes (.obj, .o)
Static Libraries (.lib, .a)
Linker
Step 4: Link Edit
Executable Code (.exe)
Shared Libraries (.dll, .so)
Loader
Step 5: Load
Run
Input
CPU
Step 6: Execute
Output
知乎 @阿基米东

gdb The GNU Debugger

- gdb name-of-executable

- The purpose of a debugger such as GDB is to allow you to see what is going on "inside" another program while it executes-or what another program was doing at the moment it crashed.
- “gcc -g ...” is a have-to.

run r	Start program execution from the beginning of the program. The command break main will get you started. Also allows basic I/O redirection.
break funtion-name break line-number b	Suspend program at specified function of line number.
s	Step to next line of code. Will step into a function.
next n	Execute next line of code. Will not enter functions.
continue c	Continue executing until next break point/watchpoint.
delete d	Delete all breakpoints, watchpoints, or catchpoints.
-q	Do not print the introductory and copyright messages.
list line-number list function l	List source code.
p variable-name	Print value stored in variable.

Command for Process Monitor--ps

F**unction:** report the snapshot of the current processes.

usage: ps option **Examples**: ps -aux , ps -ef , ps -aux | grep yourprocess, ps -ef | less

most used options:

a: display all the processes and its names;

-a: display all the processes of the terminal;

r: display the running processes;

T: display the processes for the current terminal;

-aux: display all the processes include both processes for terminal and non terminal.

-u username: display the processes of the user who is specified by usename.

-f : Do full-format listing.

System calls related to process

1. fork(): create a new process.
2. exec() family of function: replace the current process image with a new process image.
3. wait(): wait for a child process.
4. exit(): terminate a process.
5. getpid(): returns the process ID (PID) of the calling process.
6. getppid(): returns the process ID of the parent of the calling process.

fork : create a new process.

pid_t fork(void);

- create a new process by duplicating the calling process;
- Return Value:

return twice, 0 for child process and the pid of child process for parent process. If failure, -1 returned.

```
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
int main()
{
    pid_t pid;
    pid = fork();
    if (0==pid)    {
        printf("I am child process\n");

    }
    else if(pid>0)    {

        printf("I am parent process\n");

    }
    else    {
        printf("fork error\n");
    }

    exit(0);
}
```

Exec family function

int excvX(const char* path, argument)

causes the current process to abandon the program that it is running and start running the program in file path.

There are there basic system calls:

excel: the number of the arguments is not uncertain, make sure the last argument should be NULL;

execv: the argument is a array, the last element of the array is NULL;

execvp : need not specify the path for the new program, and load the new program according to the default path(/usr/bin /usr/local/bin) ;

The example for exec family function

```
char *exec_argv[4];
 exec_argv[0] = "telnet“;
 exec_argv[1] = ip;
 exec_argv[2] = port;
 exec_argv[3] = NULL;
 if (execv("/bin/telnet", exec_argv) == -1)
 {

     DoDisconnect();

     CheckError(nResult, etTelnetConnect, "Connect");

 }
```

Or: execl(“/usr/bin/ls”,”ls ”,”-al”,NULL);

exit()与_exit()

void exit(int status);

- Cause normal process terminate. Before exit the process, the process will close all the file descriptors, clean all the buffers and perform at_exit() function.
- _exit(): just close file descriptors before terminate the process.

Synchronization between parent and child processes

- pid_t wait(int *stat_loc);

Until wait return -1 and errno is ECHILD, all child processes end.

- pid_t waitpid(pid_t pid,int *stat_loc,int options);
  - pid = waitpid(-1, NULL, 0);
  - void sig_chld(int signo) {

```
void sig_chld(int signo) {
    pid_t pid;      int stat;
    while ( (pid = waitpid(-1, &stat, WNOHANG)) > 0)
//same as wait()
    printf("child %d terminated\n", pid);
    return;
}
```

  - while(waitpid(-1, NULL, WNOHANG) > 0)) ; //wait untill all children end.

```
pid_t   pid=fork();

 if(pid>0)    {
    int status=0;
    printf("Parent process\n");
    wait(&status);
    if(WIFEXITED(status))       {
       printf("child process return exit code:%d\n",WEXITSTATUS(status));
    }else if(WIFSIGNALED(status))       {
       printf("child process return signaled code:%d\n",WTERMSIG(status));
    }else if(WIFSTOPPED(status))       {
       printf("child process return stopped code:%d\n",WSTOPSIG(status));
    }else       {
       printf("other code !   \n");
    }
 }else if(pid==0)    {
    printf("i am child !\n");
 }
 printf("game is over!\n");
 return 0;
```

Experiment

0. Compile and run all the c files, and try to understand them.

In classroom:

1. Write code to let your program (named by your name&ID) sleep long enough, and then use "ps" to find your program. Take a screenshot and submit to wechat group.

Lab report:

2. Create a child process using system call fork. Then print the parent ID in parent process and print child ID in child process by function *getpid*();
3. Create multi-process using fork, where:
   - Parent process prints the sentence "Game begins";
   - Parent process wait for the execution of the child process;
   - Child process will list the files of the current directory ;
   - Parent processes prints the sentence "Game over" and then quit.
4. Complete the experiment report 3.

- Copy your codes and screenshots in report.
- Make all your figures or screenshots unique.
- Due at Nov. 4, 24:00. Please submit in time to TA's email. Any delay of submitting will get points cut.

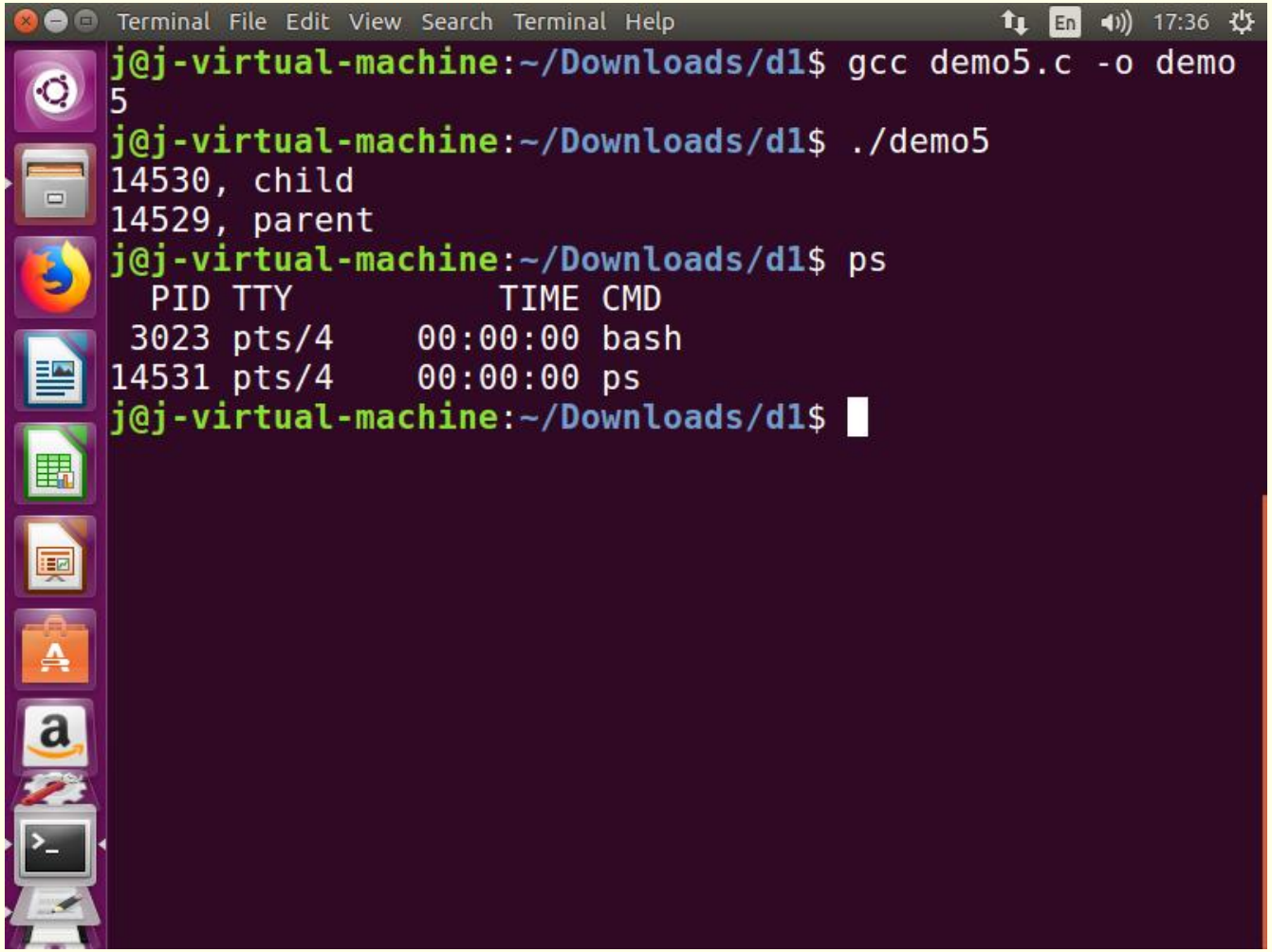
In classroom practice

```
#include <stdio.h>
#include <time.h>


int main()
{
        char string[]="Hello World by your Name-ID!";
        while (1)
        {
                printf("%s\n",string);
                sleep(1);
        }

}
```

```
root@ubuntu2204:/home/j/lab3# ls
NameID  demo6  demo6.c  demoSleep.c
root@ubuntu2204:/home/j/lab3# ps -au | grep NameID
root       123628  0.0  0.0   2776  1280 pts/1    S+
   16:33   0:00 ./NameID
root       123642  0.0  0.0   8996  2048 pts/0    S+
   16:35   0:00 grep --color=auto NameID
root@ubuntu2204:/home/j/lab3#
```

```
root@ubuntu2204:/home/j/lab3# ls
NameID  demo6  demo6.c  demoSleep.c
root@ubuntu2204:/home/j/lab3# ./NameID
Hello World by your Name-ID!
Hello World by your Name-ID!
Hello World by your Name-ID!
Hello World by your Name-ID!
Hello World by your Name-ID!
Hello World by your Name-ID!
Hello World by your Name-ID!
Hello World by your Name-ID!
Hello World by your Name-ID!
Hello World by your Name-ID!
```

Lab 1



Nothing in the world is difficult for one who sets his mind to it.
世上无难事，只怕有心人。

lab 2

Activities Terminal 10月 28 11:37

root@ubuntu2204: /home/j/lab3

```
root@ubuntu2204:/home/j/lab3# gcc demo6.c -o demo6
root@ubuntu2204:/home/j/lab3# ./demo6
4819, parent. Game begins
4820, child
total 28
drwxr-xr-x  2 root root  4096 10月 28 11:36 .
drwxr-x--- 16 j    j     4096 10月 28 11:28 ..
-rwxr-xr-x  1 root root 16184 10月 28 11:36 demo6
-rwxrwx---  1 root root   568 10月 28 10:06 demo6.c
4819, parent. Game over
root@ubuntu2204:/home/j/lab3# ls
demo6  demo6.c
root@ubuntu2204:/home/j/lab3#
```

书山有路勤为径, 学海无涯苦作舟

Diligence is the path to the mountain of knowledge, hard-working is the boat to the endless sea of learning.