

Lab1 环境配置

一、实验概述

安装ubuntu20.04, 熟悉linux系统常用命令, 配置实验环境。通过此次实验对linux系统有个初步了解。

二、实验目的

1. 安装ubuntu20.04
2. 熟悉linux常见命令和使用方式
3. 配置后续实验需要使用的实验环境

三、实验内容

1. 安装ubuntu20.04 (可直接安装或安装虚拟机)
2. 使用terminal
3. 使用指令: ls, man, pwd, cd, mkdir, rm, cp, mv, history
4. 使用指令: echo, find, cat, grep, |(pipe), >, >>, <
5. 使用指令: sudo, chmod
6. 安装vim
7. 通过gcc运行一个c程序
8. 使用指令: ps, kill, pstree
9. 使用指令: ctrl+c, ctrl+z, fg
10. 安装实验环境qemu
11. 安装riscv-gcc编译器
12. 完成课堂报告

四、实验流程及相关知识点

第一步. 安装ubuntu20.04 (可直接安装或安装虚拟机)

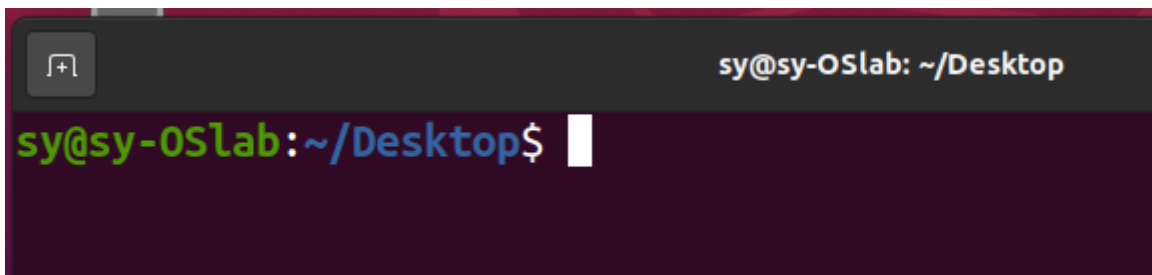
直接安装ubuntu20.04系统至计算机或参考手册在windows系统中安装虚拟机软件。

- 请注意将ubuntu系统用户名设置为你的8位学号。
- 建议使用20.04版本系统, 使用其他版本系统可能导致实验过程中部分步骤无法顺利完成。
- 语言建议使用英文, 中文路径可能导致部分实验内容无法顺利完成

第二步. 使用terminal

可以通过以下两种方式打开终端terminal

- 在Show Applications中搜索Terminal后单击打开
- 在桌面或者文件夹内点击鼠标右键, 选择Open in Terminal

A terminal window with a dark background. The title bar shows a window icon and the text 'sy@sy-OSlab: ~/Desktop'. The main area displays the prompt 'sy@sy-OSlab:~/Desktop\$' in green and blue text, followed by a white cursor.

- sy-OSlab代表当前主机名
- sy代表当前用户名
- ~/Desktop代表当前工作目录，相当于你的所有命令都是在这个目录执行的
- ~代表相对路径"/home/用户名"，相当于当前用户所属的目录
- \$代表目前用户为普通用户非管理员用户
- 通过键盘的上、下按钮可以选择历史命令
- 通过输入部分路径或文件名后点击Tab键可以自动补全

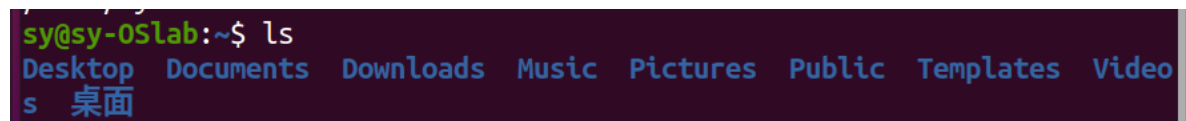
CUI vs GUI

CUI(Command User Interface)，命令行用户接口，用户通过文本命令对操作系统进行交互，如windows系统中的Command Line和我们即将使用的Linux系统的Terminal。在本课程实验中，我们主要通过CUI操作完成实验。

GUI(Graphical User Interface)，图形用户接口，用户通过对图形化的界面进行多种形式（鼠标、键盘等输入设备）的操作来与系统进行交互，如我们日常使用的电脑桌面系统及手机操作。

第三步. 使用指令: ls, man, pwd, cd, mkdir, rm, cp, mv, history

ls命令，列出当前路径下的所有文件（文件夹）

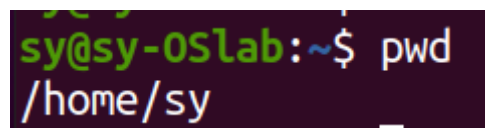
A terminal window showing the command 'ls' being executed. The output lists directories: Desktop, Documents, Downloads, Music, Pictures, Public, Templates, Video, and s. Below 's' is the Chinese word '桌面'.

请尝试"ls -l" 及 "ls -a"

man命令，帮助指令，通过man指令可以查看linux指令的使用帮助

请尝试"man ls"

pwd命令，查看当前目录完整的绝对路径

A terminal window showing the command 'pwd' being executed. The output is '/home/sy'.

cd命令，跳转工作目录

```

sy@sy-OSlab:~$ pwd
/home/sy
sy@sy-OSlab:~$ cd oslab
sy@sy-OSlab:~/oslab$ cd ..
sy@sy-OSlab:~$ pwd
/home/sy
sy@sy-OSlab:~$ cd oslab
sy@sy-OSlab:~/oslab$ cd /
sy@sy-OSlab:/$ pwd
/
sy@sy-OSlab:/$ cd ~
sy@sy-OSlab:~$ pwd
/home/sy

```

"cd ..", 跳转至上级目录

"cd 路径"跳转至路径的目录

"cd /"跳转至系统根目录, linux系统根目录为/

"cd ~"跳转至当前用户目录, 即"/home/用户名"目录

mkdir命令, 在当前目录创建文件夹

```

sy@sy-OSlab:~$ ls
Desktop    Downloads  Pictures   Templates  桌面
Documents  Music      Public     Videos
sy@sy-OSlab:~$ mkdir OSlab
sy@sy-OSlab:~$ mkdir oslab
sy@sy-OSlab:~$ ls
Desktop    Downloads  OSlab     Pictures   Templates  桌面
Documents  Music      oslab     Public     Videos

```

linux系统是严格区分大小写的, 同一个字母的大小写会作为不同的两个字母

rm命令, 删除文件或文件夹

```

sy@sy-OSlab:~$ ls
Desktop    Downloads  OSlab     Pictures   Templates  Videos
Documents  Music      oslab     Public     test.txt   桌面
sy@sy-OSlab:~$ rm test.txt
sy@sy-OSlab:~$ ls
Desktop    Downloads  OSlab     Pictures   Templates  桌面
Documents  Music      oslab     Public     Videos

```

删除文件夹时可能会碰到以下报错

```

sy@sy-OSlab:~$ rm oslab
rm: cannot remove 'oslab': Is a directory

```

这是由于文件夹与文件不同, 文件夹内可能有文件或文件夹, 因此我们需要“递归地”进行删除, 因此在删除时增加 -r 参数以递归地删除文件夹

```

sy@sy-OSlab:~$ ls
Desktop  Downloads  OSlab  Pictures  Templates  桌面
Documents Music      oslab  Public    Videos
sy@sy-OSlab:~$ rm oslab
rm: cannot remove 'oslab': Is a directory
sy@sy-OSlab:~$ rm -r oslab
sy@sy-OSlab:~$ ls
Desktop  Downloads  OSlab  Public  Videos
Documents Music      Pictures Templates 桌面

```

cp命令，复制文件或文件夹

```

sy@sy-OSlab:~/OSlab$ ls
test.txt
sy@sy-OSlab:~/OSlab$ cp test.txt test.txt.bak
sy@sy-OSlab:~/OSlab$ ls
test.txt  test.txt.bak

```

请尝试通过cp命令复制文件夹

mv命令，移动文件或文件夹，同目录下移动相当于重命名操作

```

sy@sy-OSlab:~/OSlab$ ls
test.txt  test.txt.bak
sy@sy-OSlab:~/OSlab$ mv test.txt ~
sy@sy-OSlab:~/OSlab$ ls
test.txt.bak
sy@sy-OSlab:~/OSlab$ cd ..
sy@sy-OSlab:~$ la
.bash_history  Documents  .pam_environment  Templates
.bash_logout   Downloads  Pictures           test.txt
.bashrc        .gnupg    .profile          Videos
.cache         .local    Public            .viminfo
.config        Music     .ssh              桌面
Desktop        OSlab    .sudo_as_admin_successful
sy@sy-OSlab:~$ mv test.txt testrename.txt
sy@sy-OSlab:~$ ls
Desktop  Downloads  OSlab  Public  testrename.txt  桌面
Documents Music      Pictures Templates  Videos

```

history命令，查看历史命令

第四步. 使用指令: echo, find, cat, grep, |(pipe), >, >>, <

echo命令，输出内容

```

sy@sy-OSlab:~$ echo $HOME
/home/sy
sy@sy-OSlab:~$ echo "hahaha"
hahaha

```

find命令，查找文件

```
sy@sy-OSlab:~/Desktop$ ls
a.out  hello.c  qemu-5.0.0  qemu-5.0.0.tar.xz  ucronrv
sy@sy-OSlab:~/Desktop$ find *.c
hello.c
```

可以指定通过文件名、文件类型、大小等信息进行查找

cat命令，在terminal中查看文件内容

```
sy@sy-OSlab:~/Desktop$ cat hello.c
int main(){
    while(1){
    }
    return 0;
}
```

相关的命令还有head, tail, more, less，可以实现看文件头尾，分页查看的功能

```
sy@sy-OSlab:~/Desktop$ head -n 2 hello.c
int main(){
    while(1){
sy@sy-OSlab:~/Desktop$ tail -n 2 hello.c
    return 0;
}
```

grep命令，查找文件中符合条件的字符串

```
sy@sy-OSlab:~/Desktop$ cat text
hello world
hello cse
hello os

sy@sy-OSlab:~/Desktop$ grep hello text
hello world
hello cse
hello os
```

| (pipe)操作符，将|符号前命令的输出作为|符号后命令的输入

```
sy@sy-OSlab:~/Desktop$ ls
a.out  hello.c  qemu-5.0.0  qemu-5.0.0.tar.xz  text  ucronrv
sy@sy-OSlab:~/Desktop$ ls | grep ll
hello.c
```

>, >>, < 操作符，重定向输入输出

```

sy@sy-OSlab:~/Desktop$ ls >test
sy@sy-OSlab:~/Desktop$ ls
a.out  hello.c  qemu-5.0.0  qemu-5.0.0.tar.xz  test  text  ucoreonrv
sy@sy-OSlab:~/Desktop$ cat test
a.out
hello.c
qemu-5.0.0
qemu-5.0.0.tar.xz
test
text
ucoreonrv

```

>可以将输出重定向到文件，上图中即将ls指令的结果输出到test文件中

请尝试>>, <操作符的功能

第五步. 使用指令: sudo, chmod

sudo指令，使用管理员权限执行后面的命令

```

sy@sy-OSlab:~/Desktop$ mv test /
mv: cannot move 'test' to '/test': Permission denied
sy@sy-OSlab:~/Desktop$ sudo mv test /
[sudo] password for sy:
sy@sy-OSlab:~/Desktop$ cd /
sy@sy-OSlab:/$ ls
bin      dev      lib      libx32   mnt      root     snap     sys      usr
boot     etc      lib32    lost+found  opt      run      srv      test     var
cdrom    home     lib64    media     proc     sbin     swapfile tmp

```

当我们需要执行一些指令，但是没有管理员权限无法执行时，可使用sudo指令

请尽量不要尝试"sudo rm -rf /*"

chmod指令，修改文件或文件夹的权限

通过"ls -l"指令可以查看文件的权限

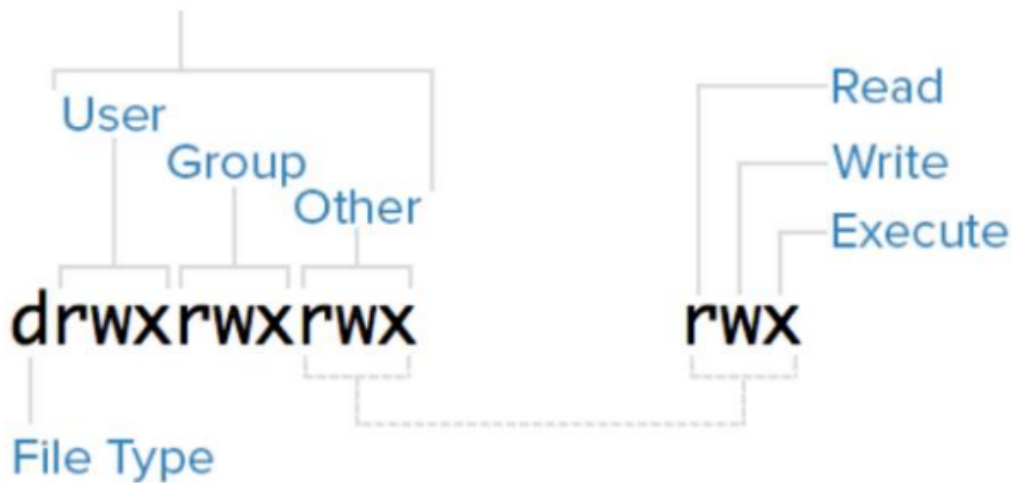
```

sy@sy-OSlab:~/Desktop$ ls -l
total 61008
-rwxrwxr-x  1 sy sy    16464 2月  12 11:30 a.out
-rw-rw-r--  1 sy sy      39 2月  12 11:30 hello.c
drwxr-xr-x 57 sy sy   12288 2月   5 15:09 qemu-5.0.0
-rw-rw-r--  1 sy sy 62426192 4月  29 2020 qemu-5.0.0.tar.xz
-rw-rw-r--  1 sy sy      32 2月  14 11:31 text
drwxrwxr-x  9 sy sy    4096 1月  21 18:31 ucoreonrv

```

上图中文件/文件夹最前方的drwxrwxr-x代表该文件/文件夹的文件权限。其所代表的含义如下图所示：

Permissions Classes



需要修改权限可以通过chmod命令

```
sy@sy-OSlab:~/Desktop$ ls -l
total 61008
-rwxrwxr-x 1 sy sy 16464 2月 12 11:30 a.out
-rw-rw-r-- 1 sy sy 39 2月 12 11:30 hello.c
drwxr-xr-x 57 sy sy 12288 2月 5 15:09 qemu-5.0.0
-rw-rw-r-- 1 sy sy 62426192 4月 29 2020 qemu-5.0.0.tar.xz
-rw-rw-r-- 1 sy sy 32 2月 14 11:31 text
drwxrwxr-x 9 sy sy 4096 1月 21 18:31 ucronrv
sy@sy-OSlab:~/Desktop$ chmod o+x text
sy@sy-OSlab:~/Desktop$ ls -l |grep text
-rw-rw-r-x 1 sy sy 32 2月 14 11:31 text
sy@sy-OSlab:~/Desktop$ chmod g-w text
sy@sy-OSlab:~/Desktop$ ls -l |grep text
-rw-r--r-x 1 sy sy 32 2月 14 11:31 text
sy@sy-OSlab:~/Desktop$ chmod u=wx text
sy@sy-OSlab:~/Desktop$ ls -l |grep text
--wxr--r-x 1 sy sy 32 2月 14 11:31 text
sy@sy-OSlab:~/Desktop$ chmod 775 text
sy@sy-OSlab:~/Desktop$ ls -l |grep text
-rwxrwxr-x 1 sy sy 32 2月 14 11:31 text
```

如上图所示，u\g\o分别代表user\group\other类别用户，+、-、=分别代表增加、减少、设置为相应的权限。

chmod 775则可以将所有组别的权限一次设置完成，数字7和5分别代表二进制111和101，二进制位上的数字分别代表rwx的相应权限，如101即代表"1可r+0不可w+1可x"，因此chmod 775即代表将该文件权限改为user组可读可写可执行，group可读可写可执行，other可读不可写可执行。

第六步. 安装vim

通过apt-get install vim 指令安装vim软件，vim是一个文件编辑器，可以通过terminal对文件进行编辑

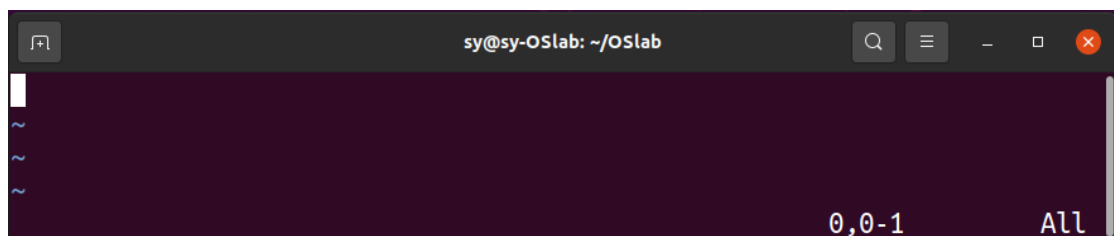

```

sy@sy-OSlab:~/Desktop$ sudo apt-get install vim
[sudo] password for sy:
Reading package lists... Done
Building dependency tree
Reading state information... Done
vim is already the newest version (2:8.1.2269-1ubuntu5.7).
The following packages were automatically installed and are no longer re
quired:
  ibverbs-providers ipxe-qemu libaio1 libcacard0 libfdt1 libibverbs1
  libiscsi7 libpmem1 librados2 librbd1 librdmacm1 libslirp0
  libspice-server1 libusbredirparser1 libvirglrenderer1
  qemu-block-extra qemu-system-common qemu-system-data qemu-system-gui
  qemu-utils seabios
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 87 not upgraded.

```

安装完成后可以通过vim指令创建或打开文件

```
sy@sy-OSlab:~/OSlab$ vim hello.c
```



通过vim打开文件后会进入上图的Command mode，vim一共有三种模式：

- Command mode: 用户所有的输入都是command而不是文本
- Insert mode: 从Command mode通过点击i键进入Insert mode，进入后可以进行文本输入；通过Esc按钮退出Insert mode回到Command mode
- Last line mode: 从Command mode通过输入冒号（shift+;）进入，terminal最后一行左侧会出现“:”，此时可以输入特殊指令，如“wq”即写入（保存）后退出（write and quit）

version 1.1
April 1st, 06
翻译: 2006-5-21

vi / vim 键盘图

Esc 命令 模式																	
~ 转换 大小写	! 外部 过滤器	@ 运行 宏	# prev ident	\$ 行尾	% 括号 匹配	^ "软" 行首	& 重复 :s	* next ident	(句首) 下一 句首	= "soft" bol down	+ 后一行 行首					
~ 跳转到 标注	1	2	3	4	5	6	7	8	9	0 "硬" 行首	- 前一行 行首	= 自动 格式化					
Q 切换到 ex模式	W 下一 单词	E 词尾	R 替换 模式	T back 'till	Y 拷贝 行	U 撤销 行内命令	I 到行首 插入	O 分段 (前)	P 粘贴 (前)	{ 段首	}	段尾					
q 录制 宏	w 下一 单词	e 词尾	r 替换 字符	t 拷贝 字符	y 拷贝 行	u 撤销 命令	i 插入 模式	o 分段 (后)	p 粘贴 (后)	[杂项]	杂项					
A 在行尾 附加	S 删除行 并插入	D 删除 至行尾	F 行内字符 向前查找	G 文尾/ 行号	H 屏幕 顶行	J 合并 两行	K 帮助	L 屏幕 底行	:	命令	" 寄存器 标识	行首/ 列					
a 附加	s 删除字符 并插入	d 删除	f 行内字符 向前查找	g 附加 命令	h ←	j ↓	k ↑	l →	:	重复 命令	' 跳转到标 注的行首	未用!					
Z 退出	X 退格	C 修改 至行末	V 可视 行模式	B 前一 单词	N 查找 上一处	M 屏幕 中间行	< 反缩进	> 缩进	?	向前 搜索							
Z 附加 命令	x 删除 (字符)	c 修改	v 可视 模式	b 前一 单词	n 查找 下一处	m 设置 标注	< 反向 缩进	> 重复 命令	.	向后 搜索							

动作 移动光标, 或者定义操作的范围

命令 直接执行的命令,
红色命令进入编辑模式

操作 后面跟随表示操作范围的指令

extra 特殊功能,
需要额外的输入

q. 后跟字符参数

w.e.b命令
小写(b): quux(foo, bar, baz);
大写(B): Quux(foo, bar, baz);

原图: www.viemu.com 翻译: fdl (linuxsir)

主要ex命令:
:w (保存), :q (退出), :q! (不保存退出)
:e f (打开文件 f),
:%s/x/y/g ('y' 全局替换 'x'),
:h (帮助 in vim), :new (新建文件 in vim),

其它重要命令:
CTRL-R: 重复 (vim),
CTRL-F/-B: 上翻/下翻,
CTRL-E/-Y: 上滚/下滚,
CTRL-V: 块可视模式 (vim only)

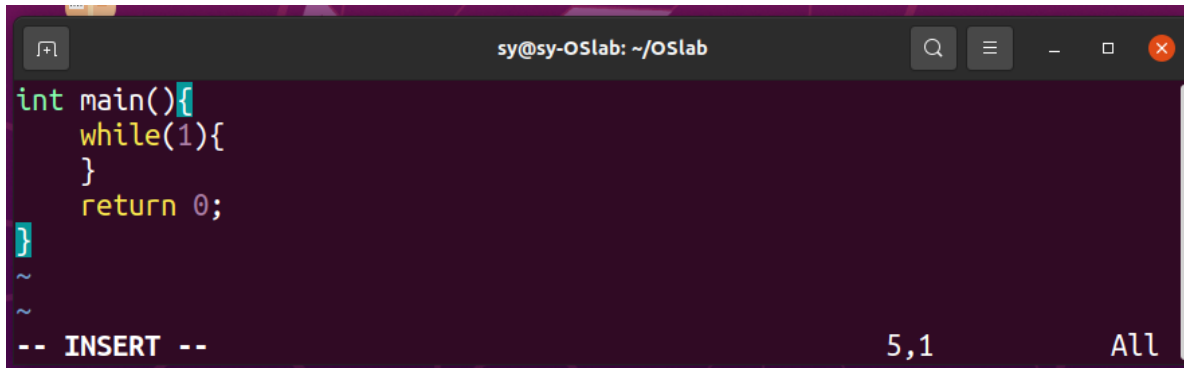
可视模式:
漫游后对选中的区域执行操作 (vim only)

备注:
(1) 在 拷贝/粘贴/删除 命令前使用 "x (x=a..z)"
使用命令的寄存器(剪贴板)
(如: "ay\$ 拷贝剩余的行内容至寄存器 'a')
(2) 命令前添加数字
多遍重复操作
(e.g.: 2p, d2w, 5i, d4j)
(3) 重复本字符在光标所在行执行操作
(dd = 删除本行, >> = 行首缩进)
(4) ZZ 保存退出, ZQ 不保存退出
(5) zt: 移动光标所在行至屏幕顶端,
zb: 底端, zz: 中间
(6) gg: 文首 (vim only),
gf: 打开光标处的文件名 (vim only)

第七步. 通过gcc运行一个c程序

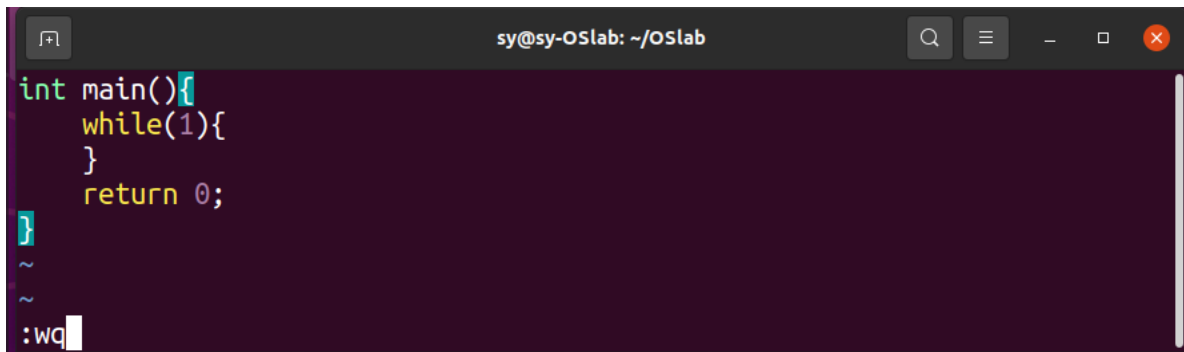
vim hello.c

通过vim创建并完成一个简单的死循环代码



```
sy@sy-OSlab: ~/OSlab
int main(){
    while(1){
    }
    return 0;
}
~
~
-- INSERT --                    5,1      All
```

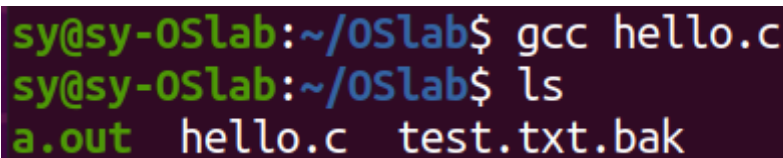
保存并退出



```
sy@sy-OSlab: ~/OSlab
int main(){
    while(1){
    }
    return 0;
}
~
~
:wq
```

gcc hello.c

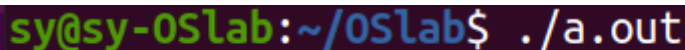
通过gcc指令对该文件进行编译并产生可执行文件，未指定输出文件名的情况下可执行文件默认文件名为a.out。更具体的gcc操作过程将在下一次实验课进行练习。本节课我们只需要能运行起一个最简单的c语言程序。



```
sy@sy-OSlab:~/OSlab$ gcc hello.c
sy@sy-OSlab:~/OSlab$ ls
a.out  hello.c  test.txt.bak
```

./a.out

通过"./a.out"指令运行a.out文件，其中.符号代表当前路径

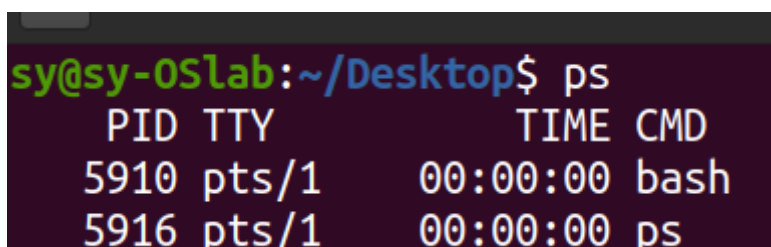


```
sy@sy-OSlab:~/OSlab$ ./a.out
```

由于我们写的是一个死循环程序，可以观察到程序执行后进入了死循环。

第八步. 使用指令: ps, kill, pstree

打开另一个terminal，执行ps指令，可以查看当前会话中的进程列表



```
sy@sy-OSlab:~/Desktop$ ps
  PID TTY          TIME CMD
 5910 pts/1        00:00:00 bash
 5916 pts/1        00:00:00 ps
```

由于之前运行的程序和新的terminal不在一个会话组，因此上图中没有显示，可以通过"ps -a"指令查看

```
sy@sy-OSlab:~/Desktop$ ps -a
  PID TTY          TIME CMD
 1584 tty2        00:00:18 Xorg
 1619 tty2        00:00:00 gnome-session-b
 5899 pts/0        00:15:39 a.out
 5954 pts/1        00:00:00 ps
```

该列表中PID为Process ID即进程号，每个进程拥有不同的进程号，一般为增序顺序分配。但进程号的数量是有限的，并且会回收再利用。

pstree指令可以查看进程之间的关系，"pstree -p"可以显示带进程号的进程树

```
├──{gnome-shell-cal}(+
└──gnome-terminal-(2726)─┬─bash(2737)─a.out+
                        └─bash(5910)─pstree+
                           └─{gnome-terminal}1(1
```

kill指令可以向进程发送中断，其中"kill -9 进程号"发送的是强制终止的信号（SIGKILL）可以用来杀死该进程号代表的进程（强制结束进程）

```
sy@sy-OSlab:~/Desktop$ ps -a
  PID TTY          TIME CMD
 1584 tty2        00:00:21 Xorg
 1619 tty2        00:00:00 gnome-session-b
 5899 pts/0        00:24:31 a.out
 5987 pts/1        00:00:00 ps
sy@sy-OSlab:~/Desktop$ kill -9 5899
sy@sy-OSlab:~/Desktop$ ps -a
  PID TTY          TIME CMD
 1584 tty2        00:00:21 Xorg
 1619 tty2        00:00:00 gnome-session-b
 5988 pts/1        00:00:00 ps
```

第九步. 使用指令: ctrl+c, ctrl+z, fg

当我们运行了一个程序无法退出，也可以不通过其他terminal发送信号来停止该进程。

ctrl+c, 终止前台进程

```
sy@sy-OSlab:~/OSlab$ ./a.out
^C
```

ctrl+z, 暂停前台进程

```
sy@sy-OSlab:~/OSlab$ ./a.out
^Z
[1]+  Stopped                  ./a.out
```

```
sy@sy-OSlab:~/OSlab$ ps -l
F S  UID      PID      PPID  C  PRI   NI     ADDR  SZ  WCHAN  TTY          TIME CMD
0 S  1000      2737      2726  0   80    0     -    4913 do_wai pts/0        00:00:00 bash
0 T  1000      6030      2737  1   80    0     -    591  do_sig pts/0        00:00:03 a.out
0 R  1000      6046      2737  0   80    0     -    5013 -      pts/0        00:00:00 ps
```

暂停的前台进程并没有被杀死，只是进入了T暂停状态。后面的课程中我们会了解到更多进程的状态。

```
sy@sy-OSlab:~/OSlab$ ./a.out
^7
[2]+  Stopped                  ./a.out
```

进程暂停时显示的号码为该进程的job号

可以通过“fg job号”命令将暂停的进程恢复到前台运行。

```
sy@sy-OSlab:~/OSlab$ fg 2
./a.out
```

第十步. 安装实验环境qemu

qemu介绍：

qemu是一个硬件模拟器，可以模拟不同架构的CPU，它甚至可以模拟不同架构的 CPU，比如说在使用 Intel X86 的 CPU 的电脑中模拟出一个 ARM 的电脑或 RISC-V 的电脑。

qemu 同时也是一个非常简单的虚拟机，给它一个硬盘镜像就可以启动一个虚拟机，并且可以定制虚拟机的配置，比如要使用的CPU、显卡啊、网络配置等，指定相应的命令行参数就可以了。它支持许多格式的磁盘镜像，包括 VirtualBox 创建的磁盘镜像文件。它同时也提供一个创建和管理磁盘镜像的工具 qemu-img。QEMU 及其工具所使用的命令行参数，直接查看其文档即可。

[About QEMU — QEMU documentation](#)

我们需要使用 Qemu 5.0.0 版本进行实验，而很多 Linux 发行版的软件包管理器默认软件源中的 Qemu 版本过低，因此 我们需要从源码手动编译安装 Qemu 模拟器。

```
# 安装编译所需的依赖包
sudo apt install autoconf automake autotools-dev curl
sudo apt install libmpc-dev libmpfr-dev libgmp-dev gawk
sudo apt install build-essential bison flex texinfo gperf
sudo apt install libtool patchutils bc zlib1g-dev libexpat-dev
sudo apt install pkg-config libglib2.0-dev libpixman-1-dev
sudo apt install git tmux python3 python3-pip
# 下载源码包
# 如果下载速度过慢可以从Blackboard下载或使用我们提供的百度网盘链接：
https://pan.baidu.com/s/1z-iWIPjxjxbdFS2Qf-NKxQ
# 提取码 8woe
```

```

wget https://download.qemu.org/qemu-5.0.0.tar.xz
# 解压
tar xvJf qemu-5.0.0.tar.xz
# 编译安装并配置 RISC-V 支持
cd qemu-5.0.0
./configure --target-list=riscv64-softmmu,riscv64-linux-user
make -j$(nproc)

# pwd 命令可以查看当前路径
pwd

# 配置环境变量
gedit ~/.bashrc

# 在文件最下面添加这三行，注意路径要替换为 自己电脑上的qemu-5.0.0 的路径，保存退出。
export PATH=$PATH:/home/oslab/Desktop/qemu-5.0.0
export PATH=$PATH:/home/oslab/Desktop/qemu-5.0.0/riscv64-softmmu
export PATH=$PATH:/home/oslab/Desktop/qemu-5.0.0/riscv64-linux-user

# 更新系统路径
source ~/.bashrc

#重启一下terminal
#测试
qemu-system-riscv64 --version
qemu-riscv64 --version

#使用下面命令可以查看是否安装成功
qemu-system-riscv64 --machine virt --nographic --bios default

```

成功会显示如下画面，按ctrl+a，然后按x退出：

```

OpensBI v0.6

  ____          ____  ____  ____
 /  _ \        /  _ \  _ \  _ \
| | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | |
 \___/ | | | | | | | | | | | | |
      | |
      | |

Platform Name      : QEMU Virt Machine
Platform HART Features : RV64ACDFIMSU
Platform Max HARTs  : 8
Current Hart       : 0
Firmware Base      : 0x80000000
Firmware Size      : 120 KB
Runtime SBI Version : 0.2

MIDELEG : 0x00000000000000222
MEDELEG : 0x000000000000b109
PMP0    : 0x0000000080000000-0x000000008001ffff (A)
PMP1    : 0x0000000000000000-0xffffffffffffffff (A,R,W,X)
QEMU: Terminated

```

第十一步. 安装riscv-gcc编译器

编译器的问题：

我们使用的计算机都是基于x86架构的。如何把程序编译到riscv64架构的汇编？这需要我们使用“目标语言为riscv64机器码的编译器”，在我们的电脑上进行**交叉编译**。

我们使用现有的riscv-gcc编译器即可

```
sudo apt install gcc-riscv64-unknown-elf
```

配置好后，在终端输入 `riscv64-unknown-elf-gcc -v` 查看安装的gcc版本, 如果输出一大堆东西且最后一行有 `gcc version 某个数字.某个数字.某个数字`，说明gcc配置成功。

第十二步. 完成课堂报告

实验课下课前以pdf格式提交至课程站点。

六、本节知识点回顾

在本次实验中，你需要了解以下知识点：

1. linux常见命令
2. 如何使用vim
3. 如何运行一个c语言程序
4. 如何查看一些进程信息

七、下一实验简单介绍

在下次实验中，我们将对c语言编程，嵌入式汇编及makefile进行介绍。