

Lab1 环境配置

一、实验概述

安装ubuntu22.04, 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ubuntu22.04
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ubuntu22.04 (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

四、实验流程及相关知识点

第一步. 安装ubuntu22.04 (可直接安装或安装虚拟机)

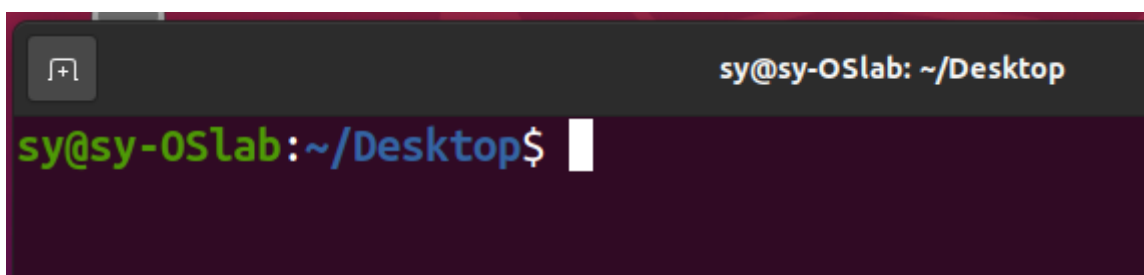
直接安装ubuntu22.04系统至计算机或参考手册在windows系统中安装虚拟机软件。

- 请注意将ubuntu系统用户名请包含你的8位学号。
- 建议使用22.04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



- 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）

```
sy@sy-OSlab:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates  Videos
s 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

```
sy@sy-OSlab:~$ pwd
/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  ucoreonrv
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  ucoreonrv
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

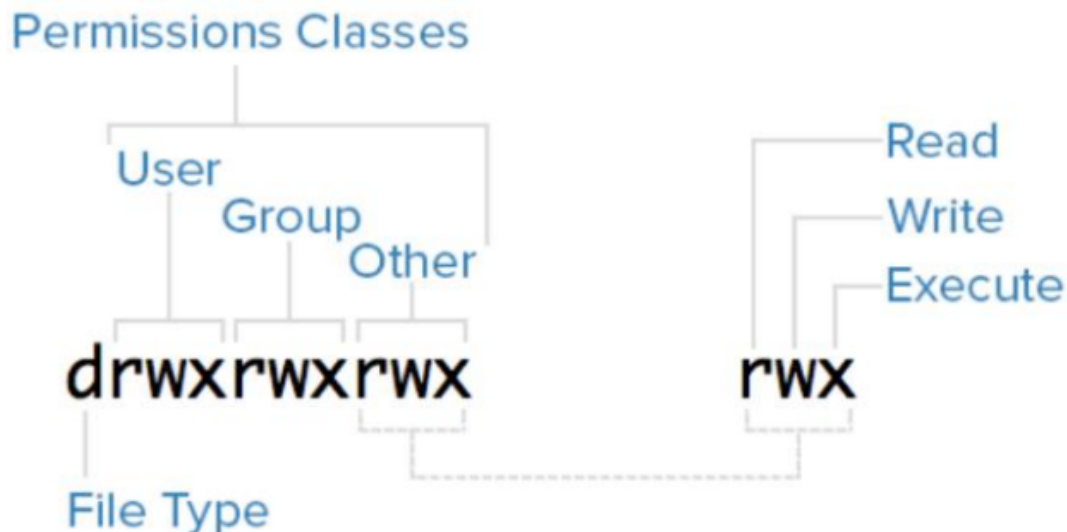
通过"ls -l"指令可以查看文件的权限

```

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：



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 ucoreonrv
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

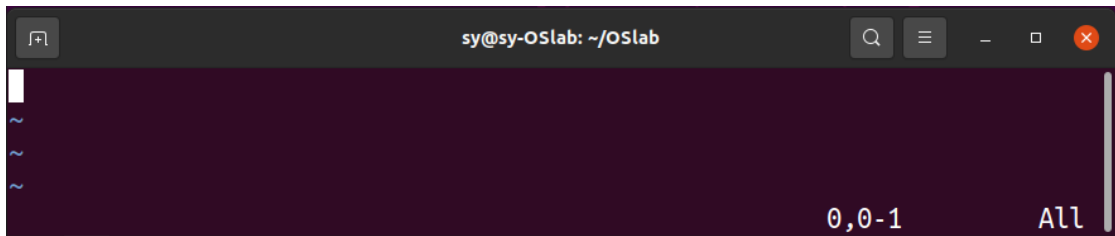
```

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 required:
  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.

```

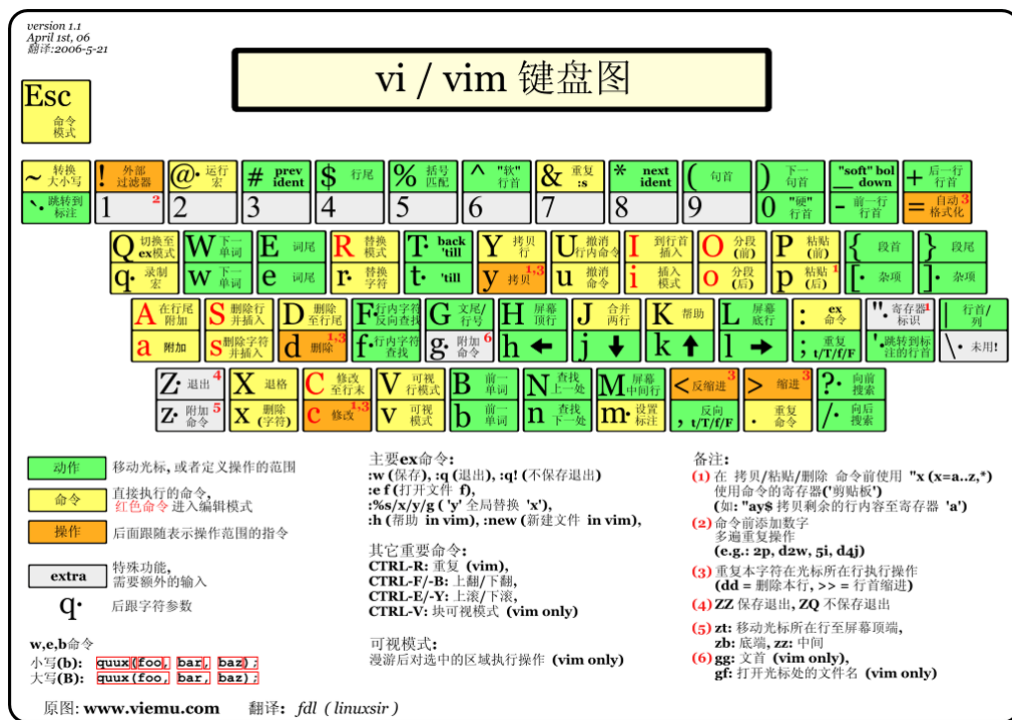
安装完成后可以通过vim指令创建或打开文件

```
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：

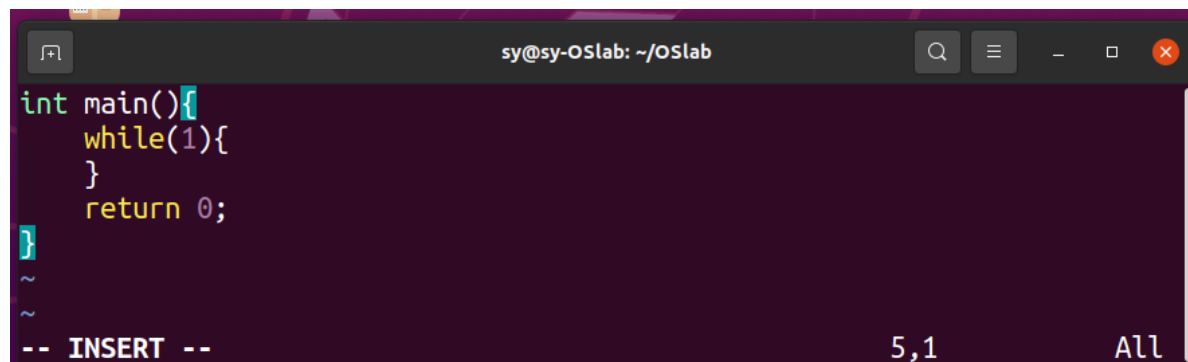
- Command mode：用户所有的输入都是command而不是文本
- Insert mode：从Command mode通过点击i键进入Insert mode，进入后可以进行文本输入；通过Esc按钮退出Insert mode回到Command mode
- Last line mode：从Command mode通过输入冒号（shift+;）进入，terminal最后一行左侧会出现“:”，此时可以输入特殊指令，如“wq”即写入（保存）后退出（write and quit）



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



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

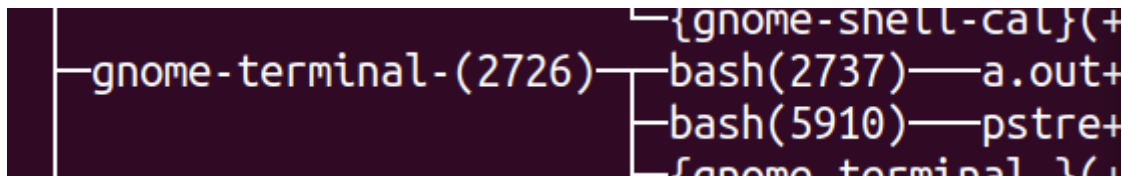
```
sy@sy-OSlab:~/Desktop$ ps
  PID TTY          TIME CMD
 5910 pts/1        00:00:00 bash
 5916 pts/1        00:00:00 ps
```

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



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. 实验环境安装

在今后的实验中，我们需要运行一个基于riscv架构的迷你操作系统，并且理解这个操作系统的代码。所以我们需要准备一个实验环境用来模拟一个riscv架构的机器，然后在这个机器上运行我们的代码。qemu就是一个硬件模拟器，可以在我们的机器上模拟出一个riscv架构的虚拟计算机。

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](#)

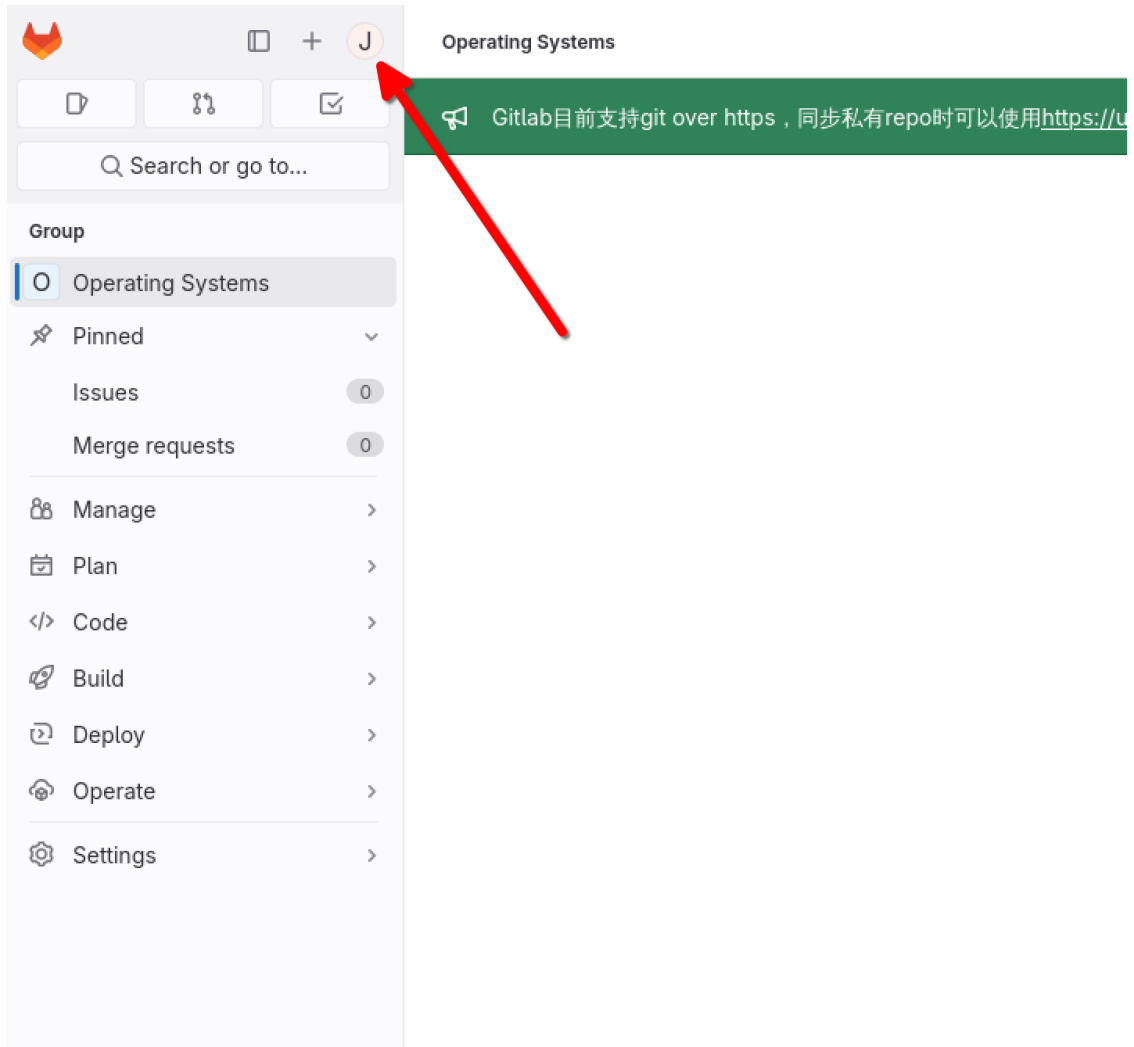
由于编译安装Qemu的过程比较复杂，本学期我们使用podman容器管理工具直接运行已经打包好实验环境的镜像进行实验。下面是podman运行实验环境的基本过程：

获取镜像的访问密钥：

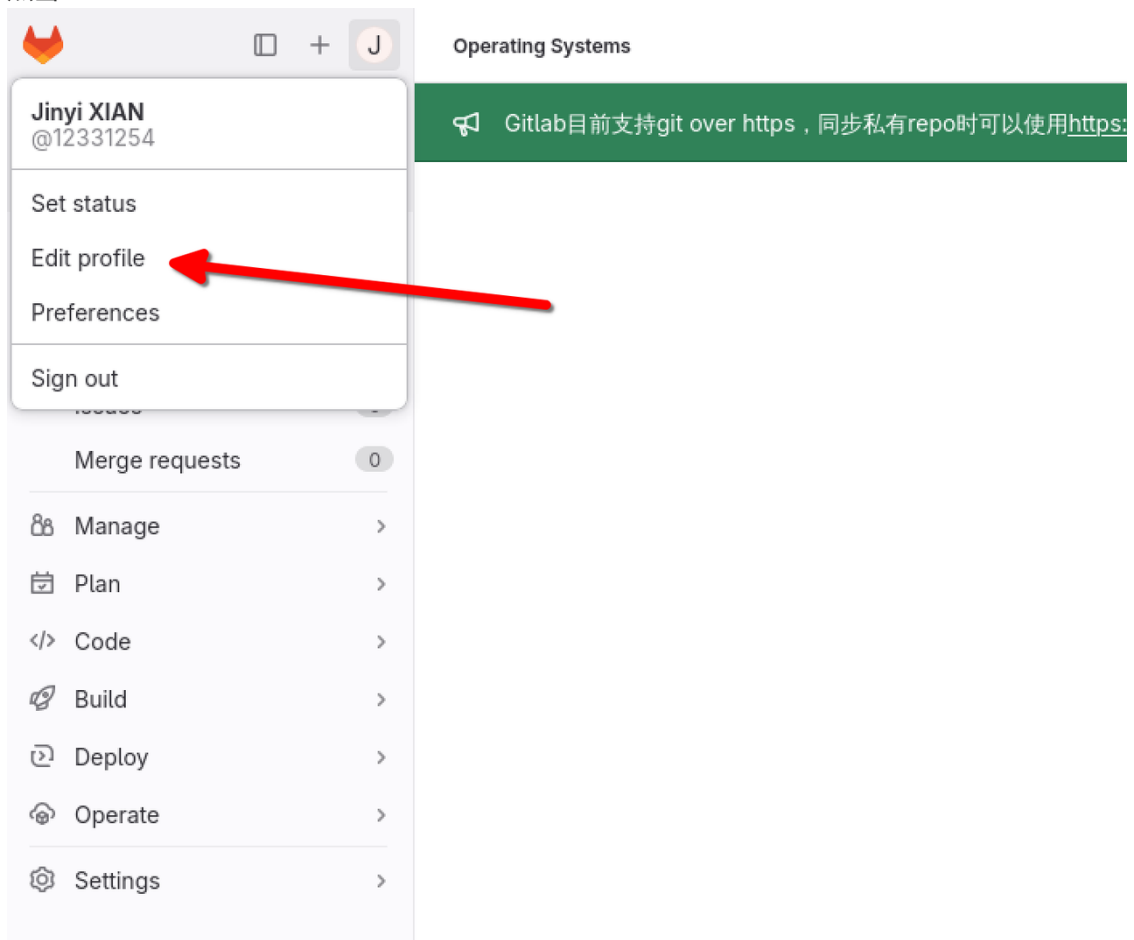
请在南科大GitLab (<https://mirrors.sustech.edu.cn/git/>) 上使用SUSTech CAS注册帐号。

之后根据以下流程生成一个访问密钥（Personal Access Token）：

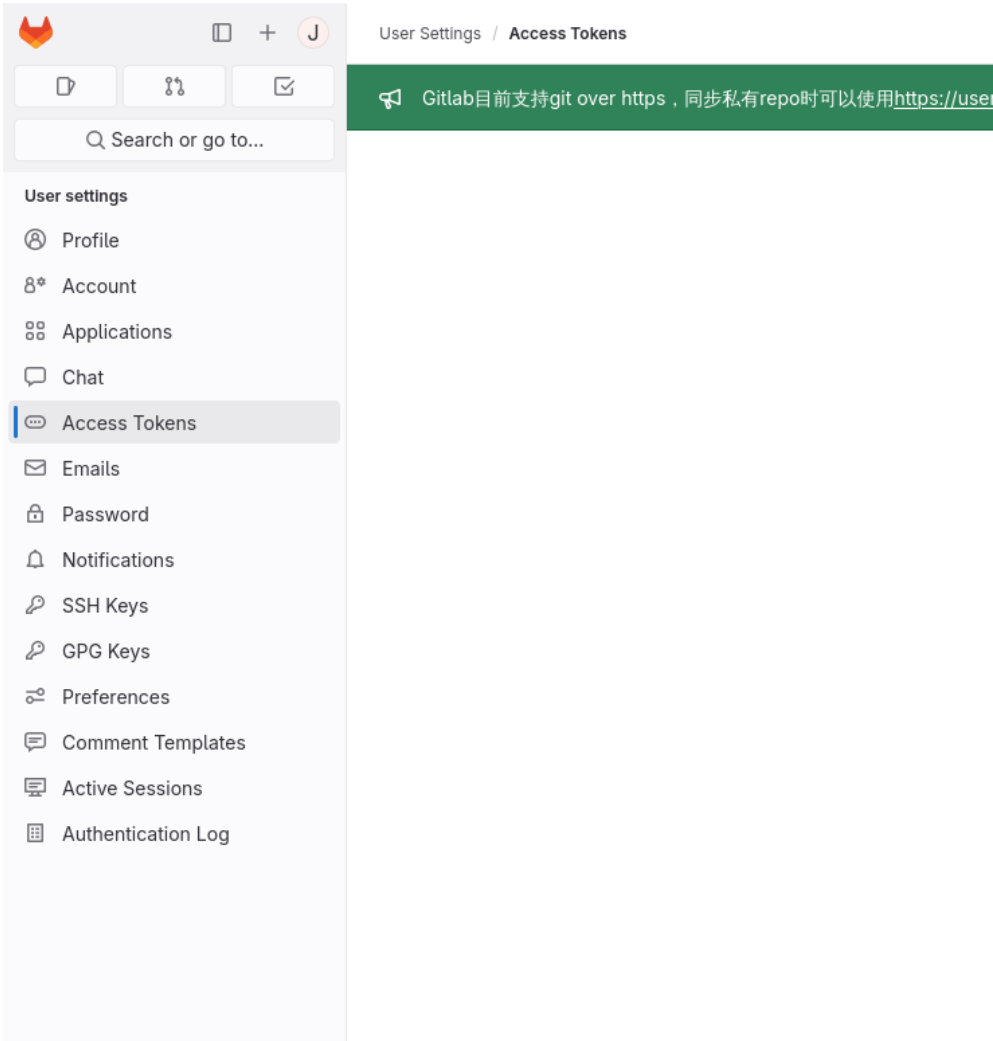
1. 先点击个人头像



2. 点击 Edit Profile



3. 在边栏找到Access Tokens



4. 点击Add New Token，输入相关信息。请将密钥过期时间设置至学期末，scopes选择read_registry。

Personal Access Tokens

You can generate a personal access token for each application you use that needs access to the GitLab API. You can also use personal access tokens to authenticate against Git over HTTP. They are the only accepted password when you have Two-Factor Authentication (2FA) enabled.

Active personal access tokens 0

Add a personal access token

Token name

registry

For example, the application using the token or the purpose of the token.

Expiration date

2024-08-01

Select scopes

Scopes set the permission levels granted to the token. [Learn more.](#)

- ☐ api
Grants complete read/write access to the API, including all groups and projects, the container registry, the dependency proxy, and the package registry.
- ☐ read_api
Grants read access to the API, including all groups and projects, the container registry, and the package registry.
- ☐ read_user
Grants read-only access to the authenticated user's profile through the /user API endpoint, which includes username, public email, and full name. Also grants access to read-only API endpoints under /users.
- ☐ create_runner
Grants create access to the runners.
- ☐ k8s_proxy
Grants permission to perform Kubernetes API calls using the agent for Kubernetes.
- ☐ read_repository
Grants read-only access to repositories on private projects using Git-over-HTTP or the Repository Files API.
- ☐ write_repository
Grants read-write access to repositories on private projects using Git-over-HTTP (not using the API).
- ☒ read_registry
Grants read-only access to container registry images on private projects.
- ☐ write_registry
Grants write access to container registry images on private projects. You need both read and write access to push images.
- ☐ ai_features
Grants access to GitLab Duo related API endpoints.

Create personal access token Cancel

Token name	Scopes	Created	Last Used	Expires	Action
This user has no active personal access tokens.					

5. 将密钥复制并保存。**注意之后将无法再在网页上获取到这个密钥，请妥善保管，不要泄漏给别人!!!**

① Your new personal access token has been created. ×

Search settings

Personal Access Tokens

You can generate a personal access token for each application you use that needs access to the GitLab API. You can also use personal access tokens to authenticate against Git over HTTP. They are the only accepted password when you have Two-Factor Authentication (2FA) enabled.

✔ Your new personal access token ×

..... 👁 📋

Make sure you save it - you won't be able to access it again.

Active personal access tokens 🔍 1 Add new token

Token name	Scopes	Created	Last Used ?	Expires	Action
registry	read_registry	Feb 19, 2024	Never	in 5 months	🗑

Feed token

Your feed token authenticates you when your RSS reader loads a personalized RSS feed or when your calendar application loads a personalized calendar. It is visible in those feed URLs. It cannot be used to access any other data.

Feed token

..... 👁 📋

Keep this token secret. Anyone who has it can read activity and issue RSS feeds or your calendar feed as if they were you. If that happens, [reset this token](#).

Incoming email token

Your incoming email token authenticates you when you create a new issue by email, and is included in your personal project-specific email addresses. It cannot be used to access any other data.

Incoming email token

..... 👁 📋

Keep this token secret. Anyone who has it can create issues as if they were you. If that happens, [reset this token](#).

安装podman

打开terminal，执行指令安装podman:

```
sudo apt install podman
```

使用podman来运行已经打包好的容器环境:

podman login用户名为学号，密码为上面获得的密钥。

```
podman login glcr.cra.ac.cn
# then input your student id and key

# pull the image, 注意这条指令需要在校内网环境执行
podman pull glcr.cra.ac.cn/operating-systems/project/kernel/ucore-tools:v0.0.4

# run the containers
podman run -v ./src --name kernel_lab -it glcr.cra.ac.cn/operating-systems/project/kernel/ucore-tools:v0.0.4 /bin/bash

# Mess around and see what you can do in the container
# for example: test the qemu version
qemu-system-riscv64 --version

# Use Ctrl + d to exit the container terminal
```

运行成功的界面:

```
bash-5.1# qemu-system-riscv64 --version
QEMU emulator version 8.1.3
Copyright (c) 2003-2023 Fabrice Bellard and the QEMU Project developers
```


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**。因此在实验镜像中，我们已经配置好了交叉编译器，可以运行指令 `riscv64-unknown-elf-gcc -v` 检查是否正常：

```
bash-5.1# riscv64-unknown-elf-gcc -v
Using built-in specs.
COLLECT_GCC=riscv64-unknown-elf-gcc
COLLECT_LTO_WRAPPER=/gnu/store/cqy1k3sid0b3f71xg8ksq04nhagv023l-gcc-cross-sans-libc-riscv64-unknown-elf-11.3.0/libexec/gcc/riscv64-unknown-elf/11.3.0/lto-wrapper
Target: riscv64-unknown-elf
Configured with:
Thread model: single
Supported LTO compression algorithms: zlib
gcc version 11.3.0 (GCC)
```

以下指令可以用于重启或删除已有的容器或镜像：

```
# restart the kernel_lab containers
podman start -a kernel_lab

# remove the containers
podman rm kernel_lab

# remove the image
podman rmi glcr.cra.ac.cn/operating-systems/project/kernel/ucore-tools:v0.0.4
# Check that you can start and stop a container.
# Find out what the command-line options above mean.
```

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。下周需要提交实验报告，因此请大家带根笔。