



中山大學

SUN YAT-SEN UNIVERSITY

实 验 报 告

课程名称：____操作系统____

姓 名：____孙广岩____

学 号：____20354242____

专业班级：____智能科学与技术 5 班____

任课教师：____吴贺俊____

____2022____年____9____月____2____日

实验报告成绩评定表

评定项目	内 容	满 分	评 分	总 分
实验态度	态度端正、遵守纪律、出勤情况	10		
实验过程	按要求完成算法设计、代码书写、 注释清晰、运行结果正确	30		
实验记录	展示讲解清楚、任务解决良好、实 验结果准确	20		
报告撰写	报告书写规范、内容条理清楚、表 达准确规范、上交及时、无抄袭， 抄袭记 0 分，提供报告供抄袭者扣 分。	40		
评语：				
<div style="float: left; width: 60%;">指导老师签字：_____</div> <div style="float: right; width: 40%; text-align: center;">年 月 日</div>				

实验一 Linux 的安装与使用

一、实验目的

1. 掌握 Linux 环境下的命令操作，熟悉 Linux 操作系统的环境和使用，记录各种测试结果。
2. 了解 LINUX 系统的安装过程，记录安装流程和界面。

二、实验内容

此次实验主要是安装 Linux 以及 Linux 系统的基本操作。

三、实验记录

1. 实验步骤

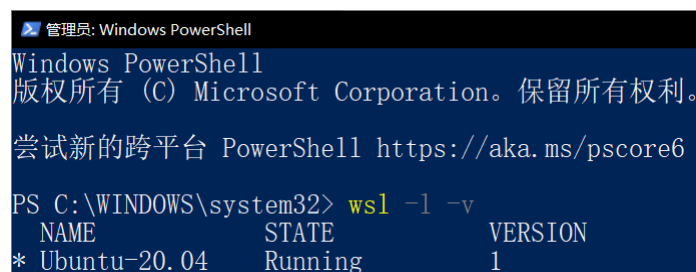
首先是安装 Linux，之后实验 Linux 系统的一些基本操作。

2. 实验记录

因为我使用的是 Windows 操作系统，所以我选择使用了 wsl，也就是 Windos Subsystem for Linux，不过新的版本 wsl2 已经使用了全新的内核，所以这里我记录一下更新的过程吧。

3. 实验结果

首先我们先看我们的 wsl 版本：



```
管理员: Windows PowerShell
Windows PowerShell
版权所有 (C) Microsoft Corporation。保留所有权利。

尝试新的跨平台 PowerShell https://aka.ms/pscore6

PS C:\WINDOWS\system32> wsl -l -v
NAME                STATE              VERSION
* Ubuntu-20.04      Running            1
```

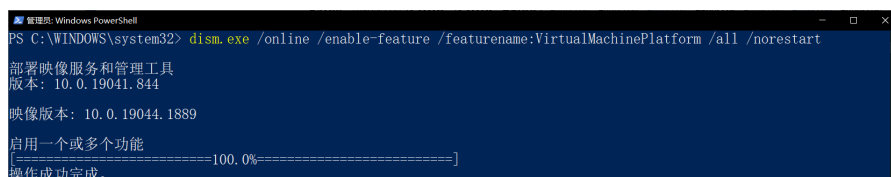
我的系统是 x86 架构的，所以下载如下的升级包：[升级包](#)

安装完成后需要安装一个 windows 功能，只需要再 powershell 中输入

```
dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart
```

执行完上述操作，更改默认分支：

```
wsl --set-default-version 2
```



```
管理员: Windows PowerShell
PS C:\WINDOWS\system32> dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart
部署映像服务和管理工具
版本: 10.0.19041.844
映像版本: 10.0.19044.1889
启用一个或多个功能
=====100.0%=====]
操作成功完成。
```

```
管理员: Windows PowerShell
PS C:\WINDOWS\system32> wsl --set-default-version 2
有关与 WSL 2 的主要区别的信息, 请访问 https://aka.ms/wsl2
操作成功完成。
PS C:\WINDOWS\system32> wsl --set-version Ubuntu20.04 2
不存在具有提供的名称的分布。
PS C:\WINDOWS\system32> wsl -l
适用于 Linux 的 Windows 子系统分发版:
Ubuntu-20.04 (默认)
PS C:\WINDOWS\system32> wsl --set-version Ubuntu-20.04 2
正在进行转换, 这可能需要几分钟时间...
有关与 WSL 2 的主要区别的信息, 请访问 https://aka.ms/wsl2
转换完成。
PS C:\WINDOWS\system32> wsl -l -v
   NAME      STATE      VERSION
* Ubuntu-20.04  Stopped      2
PS C:\WINDOWS\system32>
```

可以看出我的 wsl 升级成功, 之后就一步一步的来进行实验 ppt 中的操作。

首先是 Linux 基本的操作命令, 这里首先截图然后进行分析:

```
george@LAPTOP-NCJJCQ1J: ~ $ pwd
/home/george
george@LAPTOP-NCJJCQ1J: ~ $ ls -l
total 44
drwxr-xr-x  2 george george 4096 Aug 26 13:09 Desktop
drwxr-xr-x  2 george george 4096 Aug 26 13:09 Documents
drwxr-xr-x  2 george george 4096 Aug 26 13:25 Downloads
drwxr-xr-x  2 george george 4096 Aug 26 13:09 Music
drwxr-xr-x  2 george george 4096 Aug 26 13:09 Pictures
drwxr-xr-x  2 george george 4096 Aug 26 13:09 Public
drwxr-xr-x  2 george george 4096 Aug 26 13:09 Templates
drwxr-xr-x  2 george george 4096 Aug 26 13:09 Videos
drwxr-xr-x  2 george george 4096 Sep  2 16:44 demo
drwxr-xr-x  2 george george 4096 Sep  2 17:37 os
drwxrwxr-t  2 george george 4096 Aug 26 13:09 thinclient_drives
george@LAPTOP-NCJJCQ1J: ~ $ ls -al
total 688
drwxr-xr-x 21 george george 4096 Sep  2 17:36 .
drwxr-xr-x  3 root  root  4096 Aug 26 11:13 ..
-rw-r--r--  1 george george 358 Aug 26 13:09 .ICEAuthority
-rw-r--r--  1 george george 61 Aug 26 13:09 .Xauthority
-rw-r--r--  1 george george 2057 Aug 28 00:03 .bash_history
-rw-r--r--  1 george george 220 Aug 26 11:13 .bash_logout
-rw-r--r--  1 george george 3825 Aug 26 12:43 .bashrc
drwxrwxr-x 11 george george 4096 Aug 26 14:09 .cache
drwxrwxr-x 12 george george 4096 Aug 26 13:27 .config
drwxr-xr-x  3 george george 4096 Aug 26 13:09 .dbus
drwxr-xr-x  3 george george 4096 Aug 26 13:09 .gnupg
drwxr-xr-x  2 george george 4096 Aug 26 11:13 .landscape
drwxr-xr-x  3 george george 4096 Aug 26 13:09 .local
-rw-r--r--  1 george george 0 Sep  2 16:13 .motd_shown
drwxr-xr-x  4 george george 4096 Aug 26 13:09 .mozilla
drwxrwxrwt  2 george george 4096 Aug 26 13:09 .pcsc10
-rw-r--r--  1 george george 807 Aug 26 11:13 .profile
-rw-r--r--  1 george george 0 Aug 26 11:14 .sudo_as_admin_successful
-rw-r--r--  1 george george 9380 Sep  2 17:36 .viminfo
-rw-r--r--  1 george george 576492 Aug 26 21:14 .xorgxrdp.10.log
-rw-r--r--  1 george george 14 Aug 26 13:03 .xsession
drwxr-xr-x  2 george george 4096 Aug 26 13:09 Desktop
drwxr-xr-x  2 george george 4096 Aug 26 13:09 Documents
drwxr-xr-x  2 george george 4096 Aug 26 13:25 Downloads
drwxr-xr-x  2 george george 4096 Aug 26 13:09 Music
drwxr-xr-x  2 george george 4096 Aug 26 13:09 Pictures
drwxr-xr-x  2 george george 4096 Aug 26 13:09 Public
drwxr-xr-x  2 george george 4096 Aug 26 13:09 Templates
drwxr-xr-x  2 george george 4096 Aug 26 13:09 Videos
drwxr-xr-x  2 george george 4096 Sep  2 16:44 demo
drwxr-xr-x  2 george george 4096 Sep  2 17:37 os
drwxrwxr-t  2 george george 4096 Aug 26 13:09 thinclient_drives
george@LAPTOP-NCJJCQ1J: ~ $
```

当刚刚进入 Linux 的时候, 我们会进入到 home/[用户名]的这样一个工作目录; ls -al 中 -l 表示显示更多当前工作目录内容的详细信息, 而 -a 表示同时显示隐藏文件;

```
george@LAPTOP-NCJJCQ1J: / ~$ mkdir folder
george@LAPTOP-NCJJCQ1J: / ~$ cd /
george@LAPTOP-NCJJCQ1J: / ~$ ls -l /dev
total 0
crw-r--r-- 1 root root 10, 235 Sep  2 17:25 autofs
drwxr-xr-x 2 root root 40 Sep  2 17:30 block
drwxr-xr-x 2 root root 80 Sep  2 17:26 bsg
crw----- 1 root root 10, 234 Sep  2 17:25 btrfs-control
crw----- 1 root root 5, 1 Sep  2 17:25 console
crw----- 1 root root 10, 62 Sep  2 17:25 cpu_dma_latency
crw----- 1 root root 10, 283 Sep  2 17:25 cuse
crw-rw-rw- 1 root root 10, 61 Sep  2 17:26 dxg
lrwxrwxrwx 1 root root 13 Sep  2 17:30 fd -> /proc/self/fd
crw-rw-rw- 1 root root 1, 7 Sep  2 17:25 full
crw-rw-rw- 1 root root 10, 229 Sep  2 17:25 fuse
crw-rw-rw- 1 root root 1, 11 Sep  2 17:25 kmsg
crw----- 1 root root 10, 237 Sep  2 17:25 loop-control
brw----- 1 root root 7, 0 Sep  2 17:25 loop8
brw----- 1 root root 7, 1 Sep  2 17:25 loop1
brw----- 1 root root 7, 2 Sep  2 17:25 loop2
brw----- 1 root root 7, 3 Sep  2 17:25 loop3
brw----- 1 root root 7, 4 Sep  2 17:25 loop4
brw----- 1 root root 7, 5 Sep  2 17:25 loop5
brw----- 1 root root 7, 6 Sep  2 17:25 loop6
brw----- 1 root root 7, 7 Sep  2 17:25 loop7
```

(比较长就没有全截)

mkdir 就是创建一个空的文件夹；使用 cd / 就可以将工作目录改到根目录，ls -l [路径] 就可以看某个路径下的内容；

```
george@LAPTOP-NCJJCQ1J: / ~$ cd
george@LAPTOP-NCJJCQ1J: / ~$ pwd
/home/george
george@LAPTOP-NCJJCQ1J: / ~$ cd ../../
george@LAPTOP-NCJJCQ1J: / ~$ pwd
/
george@LAPTOP-NCJJCQ1J: / ~$
```

没有参数的 cd 会带回到用户工作空间，而使用 cd ../../ 会回到根目录，这是因为这个命令代表返回上两级目录，之前在/home/george，所以上两级回到了根目录。

之后是 Linux 文件操作命令，同样的首先是截图：

```
george@LAPTOP-NCJJCQ1J: ~$ cd subdir/
george@LAPTOP-NCJJCQ1J: ~/subdir$ date > file1
george@LAPTOP-NCJJCQ1J: ~/subdir$ cat file1
Fri Sep  2 17:54:37 CST 2022
```

a > b 在 Linux 中代表将 a 写入 b，cat 主要用于连接文件和打印，这里就是用作打印；

```
DATE(1)                                User Commands                                DATE(1)
NAME
date - print or set the system date and time
SYNOPSIS
date [OPTION]... [+FORMAT]
date [-u|--utc|--universal] [[MMDDhhmm[[CC]YY][.ss]]]
DESCRIPTION
Display the current time in the given FORMAT, or set the system date.
Mandatory arguments to long options are mandatory for short options too.
-d, --date=STRING
    display time described by STRING, not 'now'
--debug
    annotate the parsed date, and warn about questionable usage to stderr
-f, --file=DATEFILE
    like --date, once for each line of DATEFILE
-[[FMT]], --iso=8601[.FMT]
    output date/time in ISO 8601 format. FMT='date' for date only (the default), 'hours',
    'minutes', 'seconds', or 'ns' for date and time to the indicated precision.  Example:
    2006-08-24T02:34:56-0600
-R, --rfc=850
    output date and time in RFC 850 format.  Example: Mon, 14 Aug 2006 02:34:56 -0600
--rfc=3339[FMT]
    output date/time in RFC 3339 format. FMT='date', 'seconds', or 'ns' for date and time
    to the indicated precision.  Example: 2006-08-24 02:34:56-0600
-T, --reference=FILE
    display the last modification time of FILE
-s, --set=STRING
    set time described by STRING
-u, --utc, --universal
    print or set Coordinated Universal Time (UTC)
--help
    display this help and exit
--version
    output version information and exit
FORMAT controls the output.  Interpreted sequences are:
%%      a literal %
Manual page date(1) line 1 (press h for help or q to quit)
```

man 就是帮助手册；

```
george@LAPTOP-NCJJCQ1J: ~$ man date >> file1
george@LAPTOP-NCJJCQ1J: ~$ cat file1
Fri Sep  2 17:54:37 CST 2022
DATE(1)
User Commands
DATE(1)

NAME
    date - print or set the system date and time

SYNOPSIS
    date [OPTION]... [+FORMAT]
    date [-u|--utc|--universal] [MMDDhhmm[[CC]YY][.ss]]

DESCRIPTION
    Display the current time in the given FORMAT, or set the system date.

    Mandatory arguments to long options are mandatory for short options too.

    -d, --date=STRING
        display time described by STRING, not 'now'

    --debug
        annotate the parsed date, and warn about questionable usage to stderr

    -f, --file=DATEFILE
        like --date; once for each line of DATEFILE

    -I[FMT], --iso-8601[=FMT]
        output date/time in ISO 8601 format.  FMT='date' for date only (the default), 'hours',
        'minutes', 'seconds', or 'ns' for date and time to the indicated precision.  Example:
        2006-08-14T02:34:56-06:00

    -R, --rfc-email
        output date and time in RFC 5322 format.  Example: Mon, 14 Aug 2006 02:34:56 -0600

    --rfc-3339=FMT
        output date/time in RFC 3339 format.  FMT='date', 'seconds', or 'ns' for date and time
        to the indicated precision.  Example: 2006-08-14 02:34:56-06:00

    -r, --reference=FILE
        display the last modification time of FILE

    -s, --set=STRING
        set time described by STRING

    -u, --utc, --universal
        print or set Coordinated Universal Time (UTC)

    --help display this help and exit

    --version
        output version information and exit

    FORMAT controls the output.  Interpreted sequences are:
```

>> 代表接着写入，> 会覆盖掉原来的内容；

```
george@LAPTOP-NCJJCQ1J: ~$ ls -l file1
-rw-r--r-- 1 george george 6377 Sep  2 17:56 file1
george@LAPTOP-NCJJCQ1J: ~$ ls -l /usr/bin
total 154056
-rwxr-xr-x 1 root root      39 Aug 10 2019 7z
-rwxr-xr-x 1 root root      40 Aug 10 2019 7za
-rwxr-xr-x 1 root root      40 Aug 10 2019 7zr
lrwxrwxrwx 1 root root        11 Nov 29 2019 GET -> lwp-request
lrwxrwxrwx 1 root root        11 Nov 29 2019 HEAD -> lwp-request
lrwxrwxrwx 1 root root         4 Feb 17 2020 NF -> coll
lrwxrwxrwx 1 root root        11 Nov 29 2019 POST -> lwp-request
lrwxrwxrwx 1 root root         6 Apr 16 2020 Thunar -> thunar
-rwxr-xr-x 1 root root    141856 Aug 16 21:23 VGAuthService
lrwxrwxrwx 1 root root         4 Jul  6 21:53 X -> Xorg
lrwxrwxrwx 1 root root         1 Feb  8 2020 X11 -> .
-rwxr-xr-x 1 root root    2434568 Jul  6 21:53 Xephyr
-rwxr-xr-x 1 root root        274 Jul  6 21:53 Xorg
-rwxr-xr-x 1 root root    2328552 Jul  6 21:53 Xwayland
-rwxr-xr-x 1 root root     59736 Sep  5 2019 '['
-rwxr-xr-x 1 root root     31248 May 20 2020 aa-enabled
-rwxr-xr-x 1 root root     35344 May 20 2020 aa-exec
-rwxr-xr-x 1 root root      7415 Oct 26 2021 add-apt-repository
-rwxr-xr-x 1 root root     30952 Feb  7 2022 addpart
```

这里使用了/usr/bin 而不是/bin 的原因是/bin 就是一个指向/usr/bin 的链接；

```
george@LAPTOP-NCJJCQ1J: ~$ cat /var/log/syslog
total 154056
-rwxr-xr-x 1 root root 39 Aug 10 2019 7z
-rwxr-xr-x 1 root root 40 Aug 10 2019 7za
-rwxr-xr-x 1 root root 40 Aug 10 2019 7zr
lrwxrwxrwx 1 root root 11 Nov 29 2019 GET -> lwp-request
lrwxrwxrwx 1 root root 11 Nov 29 2019 HEAD -> lwp-request
lrwxrwxrwx 1 root root 4 Feb 17 2020 NF -> coll
lrwxrwxrwx 1 root root 11 Nov 29 2019 POST -> lwp-request
lrwxrwxrwx 1 root root 6 Apr 16 2020 Thunar -> thunar
-rwxr-xr-x 1 root root 141856 Aug 16 21:23 VGAuthService
lrwxrwxrwx 1 root root 4 Jul 6 21:53 X -> Xorg
-rwxrwxrwx 1 root root 1 Feb 8 2020 X11 -> .
-rwxr-xr-x 1 root root 2434568 Jul 6 21:53 Xephyr
-rwxr-xr-x 1 root root 274 Jul 6 21:53 Xorg
-rwxr-xr-x 1 root root 2328552 Jul 6 21:53 Xwayland
-rwxr-xr-x 1 root root 59736 Sep 5 2019 [
-rwxr-xr-x 1 root root 31248 May 20 2020 aa-enabled
-rwxr-xr-x 1 root root 35344 May 20 2020 aa-exec
-rwxr-xr-x 1 root root 7415 Oct 26 2021 add-apt-repository
-rwxr-xr-x 1 root root 30952 Feb 7 2022 addpart
lrwxrwxrwx 1 root root 26 Oct 20 2021 addr2line -> x86_64-linux-gnu-addr2line
-rwxr-xr-x 1 root root 274 Oct 2 2017 apg
-rwxr-xr-x 1 root root 26696 Oct 2 2017 apgbfm
-rwxr-xr-x 1 root root 2558 Dec 5 2019 apport-bug
-rwxr-xr-x 1 root root 13367 May 10 21:23 apport-cli
lrwxrwxrwx 1 root root 10 May 10 21:23 apport-collect -> apport-bug
-rwxr-xr-x 1 root root 2068 May 10 21:23 apport-unpack
-rwxr-xr-x 1 root root 14648 Feb 29 2020 appres
lrwxrwxrwx 1 root root 6 Feb 26 2020 apropos -> whatis
-rwxr-xr-x 1 root root 18824 May 24 21:08 apt
lrwxrwxrwx 1 root root 18 Oct 26 2021 apt-add-repository -> add-apt-repository
-rwxr-xr-x 1 root root 88536 May 24 21:08 apt-cache
-rwxr-xr-x 1 root root 31192 May 24 21:08 apt-cdrom
-rwxr-xr-x 1 root root 27016 May 24 21:08 apt-config
-rwxr-xr-x 1 root root 27104 May 24 21:08 apt-extracttemplates
-rwxr-xr-x 1 root root 281056 May 24 21:08 apt-ftparchive
-rwxr-xr-x 1 root root 47576 May 24 21:08 apt-get
-rwxr-xr-x 1 root root 27931 May 24 21:08 apt-key
-rwxr-xr-x 1 root root 63960 May 24 21:08 apt-mark
-rwxr-xr-x 1 root root 47504 May 24 21:08 apt-sortpkgs
-rwxr-xr-x 1 root root 1039 Dec 2 2020 aptdcon
lrwxrwxrwx 1 root root 19 Oct 20 2021 ar -> x86_64-linux-gnu-ar
-rwxr-xr-x 1 root root 39288 Sep 5 2019 arch
lrwxrwxrwx 1 root root 19 Oct 20 2021 as -> x86_64-linux-gnu-as
-rwxr-xr-x 1 root root 170496 Jul 23 2021 aspell
-rwxr-xr-x 1 root root 2044 Jul 23 2021 aspell-import
-rwsr-sr-x 1 daemon daemon 55560 Nov 13 2018 at
-rwxr-xr-x 1 root root 14640 Feb 29 2020 atobm
-rwxrwxrwx 1 root root 2 Nov 13 2018 atq -> at
-rwxrwxrwx 1 root root 2 Nov 13 2018 atrm -> at
-rwxr-xr-x 1 root root 402 Mar 21 2020 automat-visualize3
-rwxr-xr-x 1 root root 30968 Jul 6 2021 avahi-browse
lrwxrwxrwx 1 root root 12 Jul 6 2021 avahi-browse-domains -> avahi-browse
--More--
```

more 命令类似 cat，不过会以一页一页的形式显示，更方便使用者逐页阅读；

```
george@LAPTOP-NCJJCQ1J: ~/subdir$ cp file1 fa
george@LAPTOP-NCJJCQ1J: ~/subdir$ ls -l
total 16
-rw-r--r-- 1 george george 6377 Sep 2 22:08 fa
-rw-r--r-- 1 george george 6377 Sep 2 17:56 file1
george@LAPTOP-NCJJCQ1J: ~/subdir$ cd
george@LAPTOP-NCJJCQ1J: ~$ ls -l
total 48
drwxr-xr-x 2 george george 4096 Aug 26 13:09 Desktop
drwxr-xr-x 2 george george 4096 Aug 26 13:09 Documents
drwxr-xr-x 2 george george 4096 Aug 26 13:25 Downloads
drwxr-xr-x 2 george george 4096 Aug 26 13:09 Music
drwxr-xr-x 2 george george 4096 Aug 26 13:09 Pictures
drwxr-xr-x 2 george george 4096 Aug 26 13:09 Public
drwxr-xr-x 2 george george 4096 Aug 26 13:09 Templates
drwxr-xr-x 2 george george 4096 Aug 26 13:09 Videos
drwxr-xr-x 2 george george 4096 Sep 2 16:44 demo
drwxr-xr-x 2 george george 4096 Sep 2 17:37 os
drwxr-xr-x 2 george george 4096 Sep 2 22:08 subdir
drwxrwxr-t 2 george george 4096 Aug 26 13:09 thinclient_drives
```

cp 命令很好理解，就是拷贝命令，ls -l 的第一列分为几个部分，第一个字母代表文件类型，d 表示目录文件，而后面则表示每三个符号就表示一个权限，最开始的三个字母代表所有者的权限，rwx 代表读权限，写权限，可执行

权限，中间代表组用户权限，可以看到组用户是没有写权限的，最后三个字母代表其他用户权限，同样也是没有写权限的。

```
george@LAPTOP-NCJJCQ1J: ~$ grep year file1
%g      last two digits of year of ISO week number (see %G)
%G      year of ISO week number (see %V); normally useful only with %V
%j      day of year (001..366)
%q      quarter of year (1..4)
%U      week number of year, with Sunday as first day of week (00..53)
%W      week number of year, with Monday as first day of week (00..53)
%y      last two digits of year (00..99)
%Y      year
george@LAPTOP-NCJJCQ1J: ~/subdir$ find . -name "f*"
./file1
./fa
```

我这里主要列出了两个比较常用的文件操作命令，一个是 grep，用来检索文件中是否有匹配的字符串，一个是 find，也就是找到一定名称的文件的位置。

之后是程序一，Hello World， 使用 vim 来编写代码并用 gcc 编译运行：

```
george@LAPTOP-NCJJCQ1J: ~$ cat a.c
#include<stdio.h>

int main (void)
{
    printf("Hello World!\n");
}

george@LAPTOP-NCJJCQ1J: ~/os/exp1$ gcc a.c -o a
george@LAPTOP-NCJJCQ1J: ~/os/exp1$ ./a
Hello World!
```

之后是程序二，sleep 函数，使用 vim 来编写代码并用 gcc 编译运行：

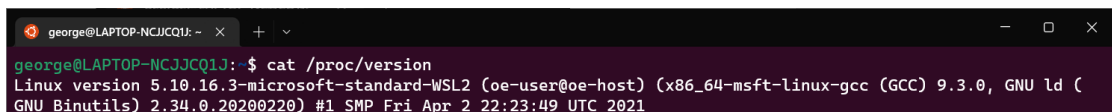
```
george@LAPTOP-NCJJCQ1J: ~$ cat hang.c
#include<stdio.h>
#include <unistd.h>

int main(int time)
{
    sleep(time);
    printf("Time for Play!\n") ;
}

george@LAPTOP-NCJJCQ1J: ~/os/exp1$ gcc hang.c -o hang
george@LAPTOP-NCJJCQ1J: ~/os/exp1$ ./hang 5
Time for Play!
george@LAPTOP-NCJJCQ1J: ~/os/exp1$
```


最后来讨论一下课后问题，首先是再 Linux 中，如何设置前、后台命令和程序的执行，我们如果没有做特殊操作，那么一般程序会默认在前台进行，想让命令在后台执行可以在命令后直接加 `&`，对于正在运行的命令，可以使用 `Ctrl+z` 来挂起，`jobs` 命令可以显示后台正在运行的进程，包括被挂起的命令，如果要将挂起的命令在后台执行，可以使用 `bg %[PID 号]`，如果想要将后台命令调到前台，那么可以使用 `fg %[PID 号]`。

查看系统的内核版本只需要使用命令 `cat /proc/version`，我的内核版本：



```
george@LAPTOP-NCJJCQ1J: ~  
george@LAPTOP-NCJJCQ1J:~$ cat /proc/version  
Linux version 5.10.16.3-microsoft-standard-WSL2 (oe-user@oe-host) (x86_64-msft-linux-gcc (GCC) 9.3.0, GNU ld (GNU Binutils) 2.34.0.20200220) #1 SMP Fri Apr 2 22:23:49 UTC 2021
```

Linus Torvalds 开发的 Linux 只是一个内核，它并不能算是一套完整的操作系统，只是有操作系统的核心功能，之后一些组织或厂商将 Linux 内核与各种软件和文档包装起来，形成 Linux 的发行版本。目前我的发行版本是 Ubuntu20.04，我也基本上只是用过 Ubuntu，也是 Linux 基础发行版 Debian 的衍生系统，其他的用的人比较多的主要有 CentOS，它是从红帽 Linux 重新构建而来，是一款企业级的 Linux 发行版。

我对 Linux 的认识主要就是两点吧，一个是开源，一个是灵活。在之前操作系统的选择不是很多，有很多都要么是比较昂贵，要不就是没有开放源代码，要么就是只能运行在特定的设备上，那么 MINIX 操作系统是一个教学用操作系统，但是它功能比较简单，于是 Linus Torvalds 就在大学期间写出了初期的 Linux 系统，并且将它全部的源代码开源，Linux 系统的开源模式之后也获得了很大的成功，这样的模型可以让 Linux 在世界各地广为传播，并可以用在各种软件项目，所以开源真的是构建生态非常重要的。同时 Linux 系统有很高的自由度，这也让更多的程序员，黑客愿意在 Linux 上进行开发，所以基于这两点也让 Linux 成为了现在最流行的系统。

四、总结与讨论

本次的实验是比较基础的，因为之前有使用 Linux 的经验所以这次的实验还是比较简单的，我也简单的写了一些别的代码来进行 gcc 的实验，Linux 的功能非常强大，技巧也非常多，希望之后可以多多学习来更加精进自己命令行的使用。

五、附：程序模块的源代码

a. c

```
1. #include<stdio.h>
2. int main (void)
3. {
4.     printf("Hello World!\n");
5. }
```

hang. c

```
1. #include<stdio.h>
2. #include <unistd.h>
3.
4. int main(int time){
5.     sleep(time);
6.     printf("Time for Play!\n") ;
7. }
```

实验二 安装 ROS

一、实验目的

1. 搭建 ROS 环境，为后续实验做准备。

二、实验内容

此次实验主要是安装 ROS 以及运行一些简单的示例程序，之前机器人课还是使用过 ROS 的，不过那个是课上直接发的镜像，所以这个就自己安装一下到我的 ws12 上面。

三、实验记录

1. 实验步骤

按照 ROS 官方给出的步骤的安装，之后运行一些简单的示例程序。

2. 实验结果

- (1) 首先配置 ubuntu 的“软件和更新”，允许安装不经认证的软件，这里使用了命令行来实现，并查看现在有的 repo:

```
george@LAPTOP-NCJJCQ1J: ~$ sudo add-apt-repository universe
'universe' distribution component is already enabled for all sources.
george@LAPTOP-NCJJCQ1J: ~$ sudo add-apt-repository multiverse
'multiverse' distribution component is already enabled for all sources.
george@LAPTOP-NCJJCQ1J: ~$ sudo add-apt-repository restricted
'restricted' distribution component is already enabled for all sources.
george@LAPTOP-NCJJCQ1J: ~$ grep ^deb /etc/apt/sources.list
deb http://archive.ubuntu.com/ubuntu/ focal main restricted
deb http://archive.ubuntu.com/ubuntu/ focal-updates main restricted
deb http://archive.ubuntu.com/ubuntu/ focal universe
deb http://archive.ubuntu.com/ubuntu/ focal-updates universe
deb http://archive.ubuntu.com/ubuntu/ focal multiverse
deb http://archive.ubuntu.com/ubuntu/ focal-updates multiverse
deb http://archive.ubuntu.com/ubuntu/ focal-backports main restricted universe multiverse
deb http://security.ubuntu.com/ubuntu/ focal-security main restricted
deb http://security.ubuntu.com/ubuntu/ focal-security universe
deb http://security.ubuntu.com/ubuntu/ focal-security multiverse
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ focal main restricted universe multiverse
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ focal-updates main restricted universe multiverse
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ focal-backports main restricted universe multiverse
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ focal-security main restricted universe multiverse
```

- (2) 之后使用 PPT 中提供的命令添加 ROS 软件源:

```
george@LAPTOP-NCJJCQ1J: ~$ sudo sh -c '. /etc/lsb-release && echo "deb http://mirrors.tuna.tsinghua.edu.cn/ros/ubuntu/ `lsb_release -cs` main" > /etc/apt/sources.list.d/ros-latest.list'
george@LAPTOP-NCJJCQ1J: ~$ sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
Executing: /tmp/apt-key-gpghome.HCT24qugai/gpg.1.sh --keyserver hkp://keyserver.ubuntu.com:80 --recv-key C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
gpg: key F42ED6FBAB17C654: public key "Open Robotics <info@osrfoundation.org>" imported
gpg: Total number processed: 1
gpg: imported: 1
```

(3) 然后使用下列命令来安装 ROS:

```
sudo apt update

sudo apt install ros-noetic-desktop-full

echo "source /opt/ros/noetic/setup.bash" >> ~/.bashrc

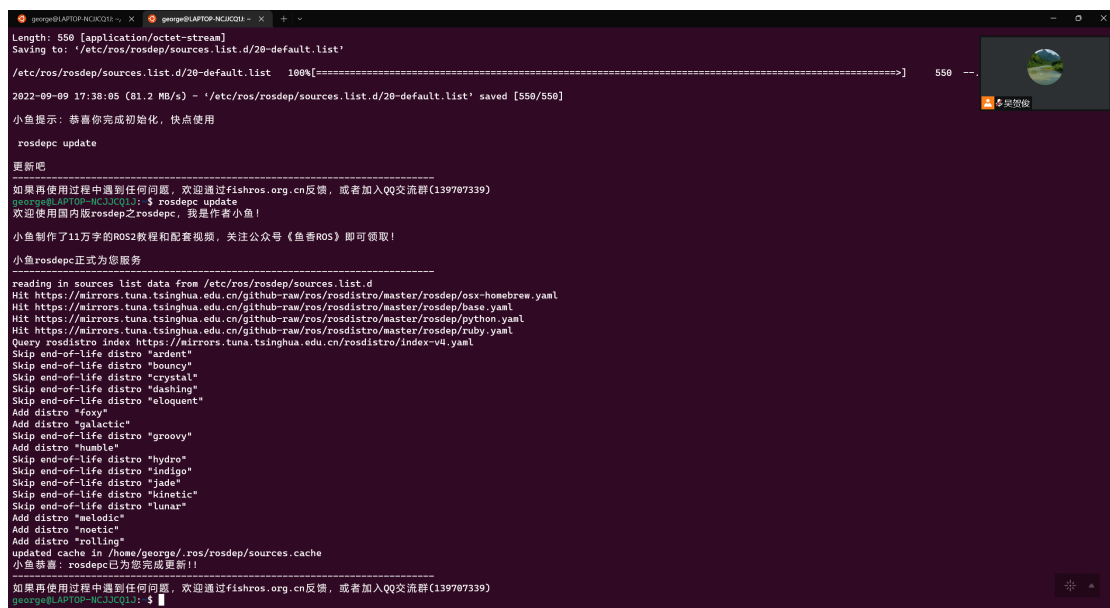
source ~/.bashrc
```

(4) 之后初始化 rosdep(这里使用了 china 版)

```
sudo pip install rosdep

sudo rosdep init

rosdep update
```



```
Length: 550 [application/octet-stream]
Saving to: '/etc/ros/rosdep/sources.list.d/20-default.list'

/etc/ros/rosdep/sources.list.d/20-default.list 100%[=====] 550 ---
2022-09-09 17:38:05 (81.2 MB/s) - '/etc/ros/rosdep/sources.list.d/20-default.list' saved [550/550]

小鱼提示: 恭喜你完成初始化, 快点使用

rosdep update

更新吧

如果在使用过程中遇到任何问题, 欢迎通过fishros.org.cn反馈, 或者加入QQ交流群(139787339)
george@LAPTOP-NCJ3CQ13:~$ rosdep update
欢迎使用国内版rosdep之rosdep, 我是作者小鱼!

小鱼制作了11万字的ROS2教程和配套视频, 关注公众号《鱼香ROS》即可领取!

小鱼rosdep正式为您服务

reading in sources list data from /etc/ros/rosdep/sources.list.d
Hit https://mirrors.tuna.tsinghua.edu.cn/github-raw/ros/rosdistro/master/rosdep/osx-homebrew.yaml
Hit https://mirrors.tuna.tsinghua.edu.cn/github-raw/ros/rosdistro/master/rosdep/base.yaml
Hit https://mirrors.tuna.tsinghua.edu.cn/github-raw/ros/rosdistro/master/rosdep/python.yaml
Hit https://mirrors.tuna.tsinghua.edu.cn/github-raw/ros/rosdistro/master/rosdep/ruby.yaml
Query rosdistro index https://mirrors.tuna.tsinghua.edu.cn/rosdistro/index-v4.yaml
Skip end-of-life distro "ardent"
Skip end-of-life distro "bouncy"
Skip end-of-life distro "crystal"
Skip end-of-life distro "dashing"
Skip end-of-life distro "eloquent"
Add distro "foxy"
Add distro "galactic"
Skip end-of-life distro "groovy"
Add distro "humble"
Skip end-of-life distro "hydro"
Skip end-of-life distro "indigo"
Skip end-of-life distro "jade"
Skip end-of-life distro "kinetic"
Skip end-of-life distro "lunar"
Add distro "melodic"
Add distro "noetic"
Add distro "rolling"
updated cache in /home/george/.ros/rosdep/sources.cache
小鱼恭喜: rosdep已为您完成更新!!

如果在使用过程中遇到任何问题, 欢迎通过fishros.org.cn反馈, 或者加入QQ交流群(139787339)
george@LAPTOP-NCJ3CQ13:~$
```

可以看到安装成功。

3. CPU 模拟指令执行 (选做)

CPU 通过遵循称为“获取、解码和执行”的过程来工作。CPU 从内存中取出一条指令, 解码这条指令, 然后执行它。这里我简单的使用了一个 Python 程序来去简单的重现一下这个 PPT 中 CPU 指令的执行过程。

创建了一个 ALU 类, 来实现简单的相加操作 (当然也可以加入其他的操作, 这里我还没有加入), 还有 CPU 类, 这个是最主要的操作 fetch-decode-execute 进行的地方, 不同的模块我进行了一定程度上的抽象, 但基本的流程还是和 CPU 运行机制一致的。

代码已在最后一部分附上了, 这是一些运行的结果:

```
▶ [3] c = CPU()
      c.cycle(Memory)

... Register Value: 3
      Control Unit: LOAD 10
      Fetch Complete
      Decode Complete
      Excute Complete
      Register Value: 4
      Control Unit: ADD 11
      Fetch Complete
      Decode Complete
      Excute Complete
      Register Value: 5
      Control Unit: STORE 12
      Fetch Complete
      Decode Complete
      Excute Complete

[4] c.accumulator

... 5

[5] new_Memory

... [{10: 2}, {11: 3}, {'LOAD': 10}, {'ADD': 11}, {'STORE': 12}, {12: 5}]
```

四、总结与讨论

本次的实验也比较简单，之前机器人课程是使用过 ROS 的，这次安装其实也非常简单，就一步一步来就可以了，没有什么难度。

CPU 的模拟指令的实现我也是参考 PPT 来写的，感觉还是非常的有趣的，事实上我写的 Python 是非常简单且基础的，目前也只能支持 PPT 中出现了三种操作，而且还有不少可以优化的部分，之后也许也可以在这个基础上进行进行扩展，来去支持更丰富的功能。

经过这两次的操作系统实验，主要还是在为后面的实验搭建环境，打下基础，我也非常期待之后更加丰富的实验。

五、附代码 ([Github 地址](#))

```
1. # 简单的 Memory 的实现
2. Memory = [{10: 2}, {11: 3}, {'LOAD':10}, {'ADD':11}, {'STORE':12}]
3. # 新的 Memory, 为了防止此次中变长 list 导致运行错误
4. new_Memory = [{10: 2}, {11: 3}, {'LOAD':10}, {'ADD':11}, {'STORE':12}
5. ]
6. class ALU(object):
7.     def __init__(self):
8.         self.n1 = None
9.         self.n2 = None
10.        self.operation = None
11.    def calculate(self):
12.        if self.operation == 'ADD':
13.            return self.n1 + self.n2
14.
15.
16. class CPU(object):
17.     def __init__(self):
18.
19.         # 简单的模拟寄存器
20.         self.register = 0
21.         self.control_unit = None
22.         self.operation = None
23.         self.accumulator = None
24.         self.alu = ALU()
25.
26.     def fetch(self, cmd):
27.         for i in range(self.register, len(cmd)):
28.
29.             key = list(Memory[i].keys())[0]
30.
31.             # 如果是储存的数值就会跳过
32.             if key == 'LOAD' or key == 'ADD' or key == 'STORE':
33.                 self.control_unit = [key, cmd[i][key]]
34.                 self.register += 1
35.                 break
36.             else:
37.                 self.register += 1
38.
39.         print('Register Value:', self.register)
40.         print('Control Unit:', self.control_unit[0], self.control_unit[1])
41.         print('Fetch Complete')
```

```

42.     def decode(self):
43.         # 在 Python 中这个比较难以体现 但这里就比较简化了
44.         if self.control_unit[0] == 'LOAD':
45.             self.operation = 'LOAD'
46.         elif self.control_unit[0] == 'ADD':
47.             self.operation = 'ADD'
48.         elif self.control_unit[0] == 'STORE':
49.             self.operation = 'STORE'
50.
51.         print('Decode Complete')
52.
53.     def get_value(self, cmd, address):
54.         for i in range(len(cmd)):
55.             if list(Memory[i].keys())[0] == address:
56.                 return Memory[i][address]
57.
58.     def execute(self, cmd):
59.         if self.operation == 'LOAD':
60.             # 取出地址
61.             address = self.control_unit[1]
62.             value = self.get_value(cmd, address)
63.             self.accumulator = value
64.         elif self.operation == 'ADD':
65.             self.alu.operation = 'ADD'
66.             self.alu.n1 = self.accumulator
67.             # 取出地址
68.             address = self.control_unit[1]
69.             value = self.get_value(cmd, address)
70.             self.accumulator = value
71.             self.alu.n2 = self.accumulator
72.             self.accumulator = self.alu.calculate()
73.
74.         elif self.operation == 'STORE':
75.             value = self.accumulator
76.             new_Memory.append({self.control_unit[1]: value})
77.
78.         print('Excute Complete')
79.
80.     def cycle(self, cmd):
81.         while(self.register <= len(cmd)-1):
82.             # fetch
83.             self.fetch(cmd)
84.             self.decode()
85.             self.execute(cmd)

```