实验一：安装Linux并学会简单的使用Linux和Windows命令

Linux部分

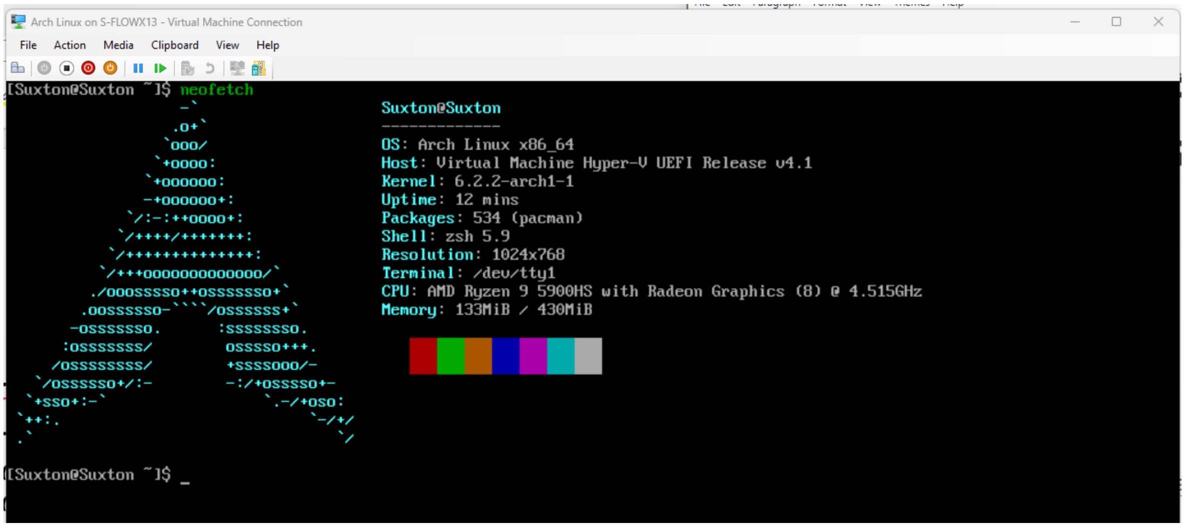
1. **安装Linux**

   在众多的发行版里面，我选择了Arch Linux，因为这个发行版是滚动更新的，可以时时刻刻使用最新的内核。

   我使用的虚拟环境是Windows专业版自带的Hyper-V工具，接下来的演示都是基于Hyper-V虚拟机。

   由于Arch Linux安装的过程过于繁琐，而且我在实验之前就已经完成了安装，所以这里我只放一张安装之后的截图。

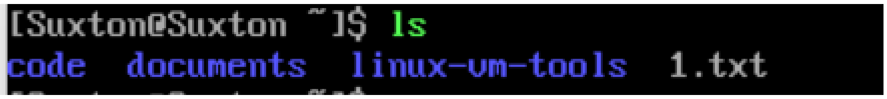
   （因为图形化界面没什么用，我就没安装桌面环境。）



```
[Suxton@Suxton ~]$ neofetch
                   -`                     Suxton@Suxton
                  .o+`                    -------------
                 `ooo/                    OS: Arch Linux x86_64
                `+oooo:                   Host: Virtual Machine Hyper-V UEFI Release v4.1
               `+oooooo:                  Kernel: 6.2.2-arch1-1
               -+oooooo+:                 Uptime: 12 mins
             `/:-:++oooo+:                Packages: 534 (pacman)
            `/++++/+++++++:               Shell: zsh 5.9
           `/++++++++++++++:              Resolution: 1024x768
          `/+++ooooooooooooo/`            Terminal: /dev/tty1
         ./ooosssso++osssssso+`           CPU: AMD Ryzen 9 5900HS with Radeon Graphics (8) @ 4.515GHz
        .oossssso-````/ossssss+`          Memory: 133MiB / 430MiB
       -osssssso.      :ssssssso.
      :osssssss/        osssso+++.        🟥🟩🟧🟦🟪🟦⬜
     /ossssssss/        +ssssooo/-
   `/ossssso+/:-        -:/+osssso+-
  `+sso+:-`                 `.-/+oso:
 `++:.                           `-/+/
 .`                                 `/

[Suxton@Suxton ~]$ _
```

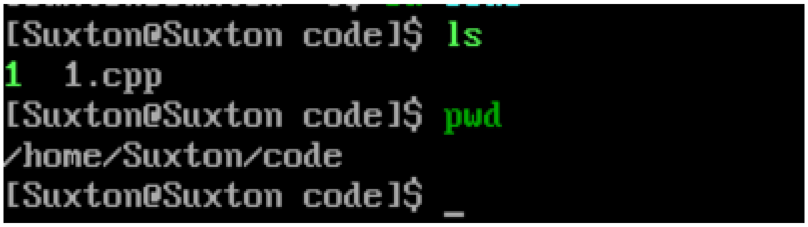
2. **熟悉Linux系统常用的命令**

1. ls 命令：用于展示当前目录下的所有的目录和文件，下面的截图展示了我的家目录下的所有文件和目录。



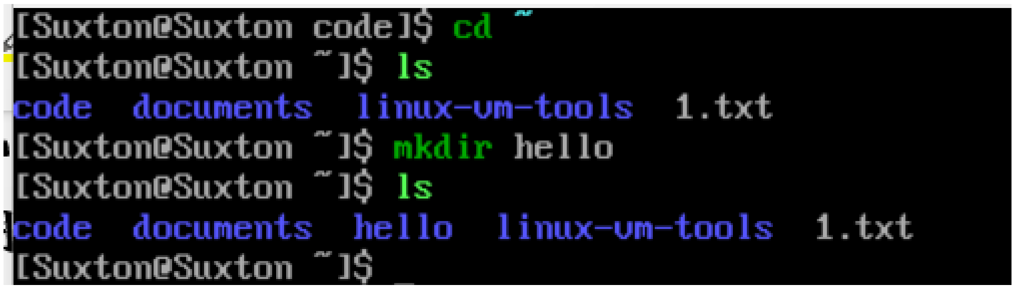
```
[Suxton@Suxton ~]$ ls
code  documents  linux-vm-tools  1.txt
```

2. cd 命令：用于选择一个目录，下面的截图中我选择了我家目录下的code目录。



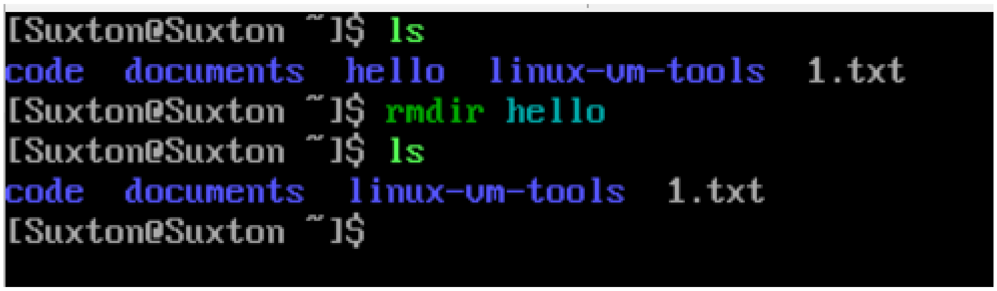
```
[Suxton@Suxton code]$ ls
1  1.cpp
[Suxton@Suxton code]$ pwd
/home/Suxton/code
[Suxton@Suxton code]$ _
```

3. mkdir 命令：在指定的位置创建一个文件夹，下面的截图在我的家目录下创建了一个hello目录。



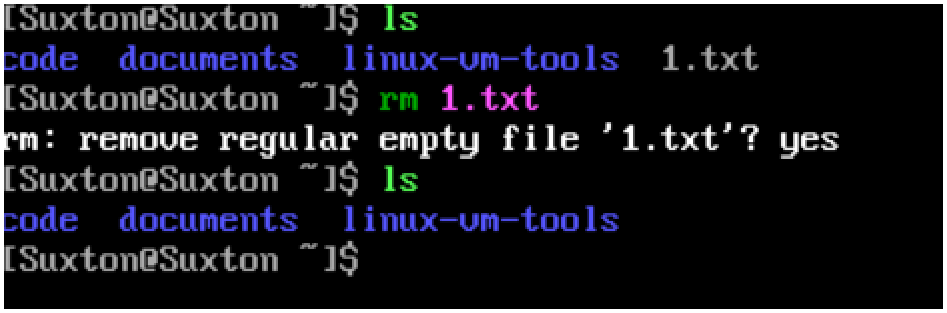
```
[Suxton@Suxton code]$ cd ~
[Suxton@Suxton ~]$ ls
code  documents  linux-vm-tools  1.txt
[Suxton@Suxton ~]$ mkdir hello
[Suxton@Suxton ~]$ ls
code  documents  hello  linux-vm-tools  1.txt
[Suxton@Suxton ~]$ _
```

4. rmdir 命令：用于删除一个空的目录，下面的截图中我把刚刚创建的hello目录删掉。



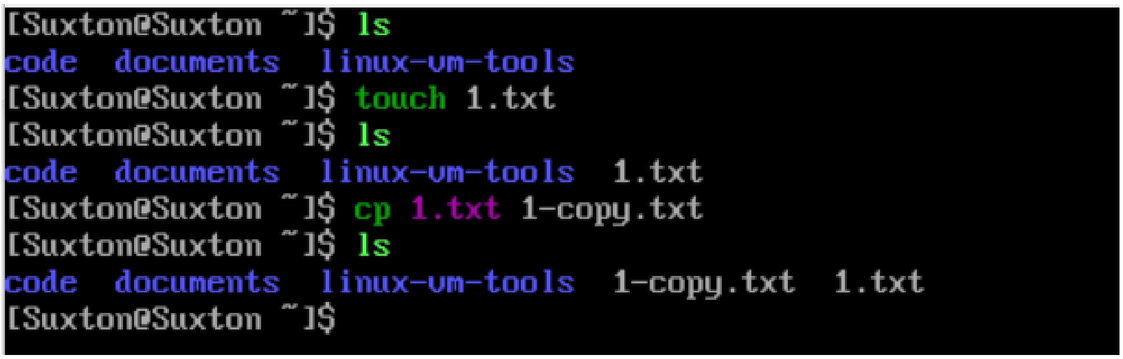
```
[Suxton@Suxton ~]$ ls
code  documents  hello  linux-vm-tools  1.txt
[Suxton@Suxton ~]$ rmdir hello
[Suxton@Suxton ~]$ ls
code  documents  linux-vm-tools  1.txt
[Suxton@Suxton ~]$
```

5. rm 命令：用于删除文件或目录（加-r），下面的截图中我把我家目录的1.txt删除了。



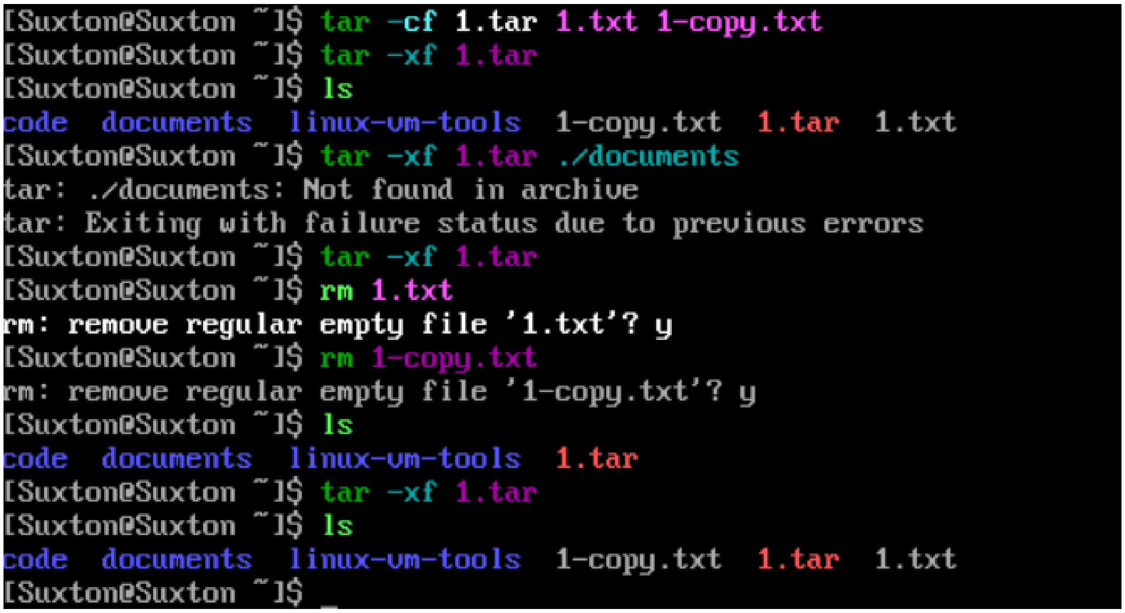
```
[Suxton@Suxton ~]$ ls
code  documents  linux-vm-tools  1.txt
[Suxton@Suxton ~]$ rm 1.txt
rm: remove regular empty file '1.txt'? yes
[Suxton@Suxton ~]$ ls
code  documents  linux-vm-tools
[Suxton@Suxton ~]$
```

6. cp 命令：用于复制一个文件，第一个参数是源文件，第二个参数是目标文件名（可以加上目录，默认为当前文件夹）。下面我先创建了一个1.txt，然后把它复制了一遍。



```
[Suxton@Suxton ~]$ ls
code  documents  linux-vm-tools
[Suxton@Suxton ~]$ touch 1.txt
[Suxton@Suxton ~]$ ls
code  documents  linux-vm-tools  1.txt
[Suxton@Suxton ~]$ cp 1.txt 1-copy.txt
[Suxton@Suxton ~]$ ls
code  documents  linux-vm-tools  1-copy.txt  1.txt
[Suxton@Suxton ~]$
```

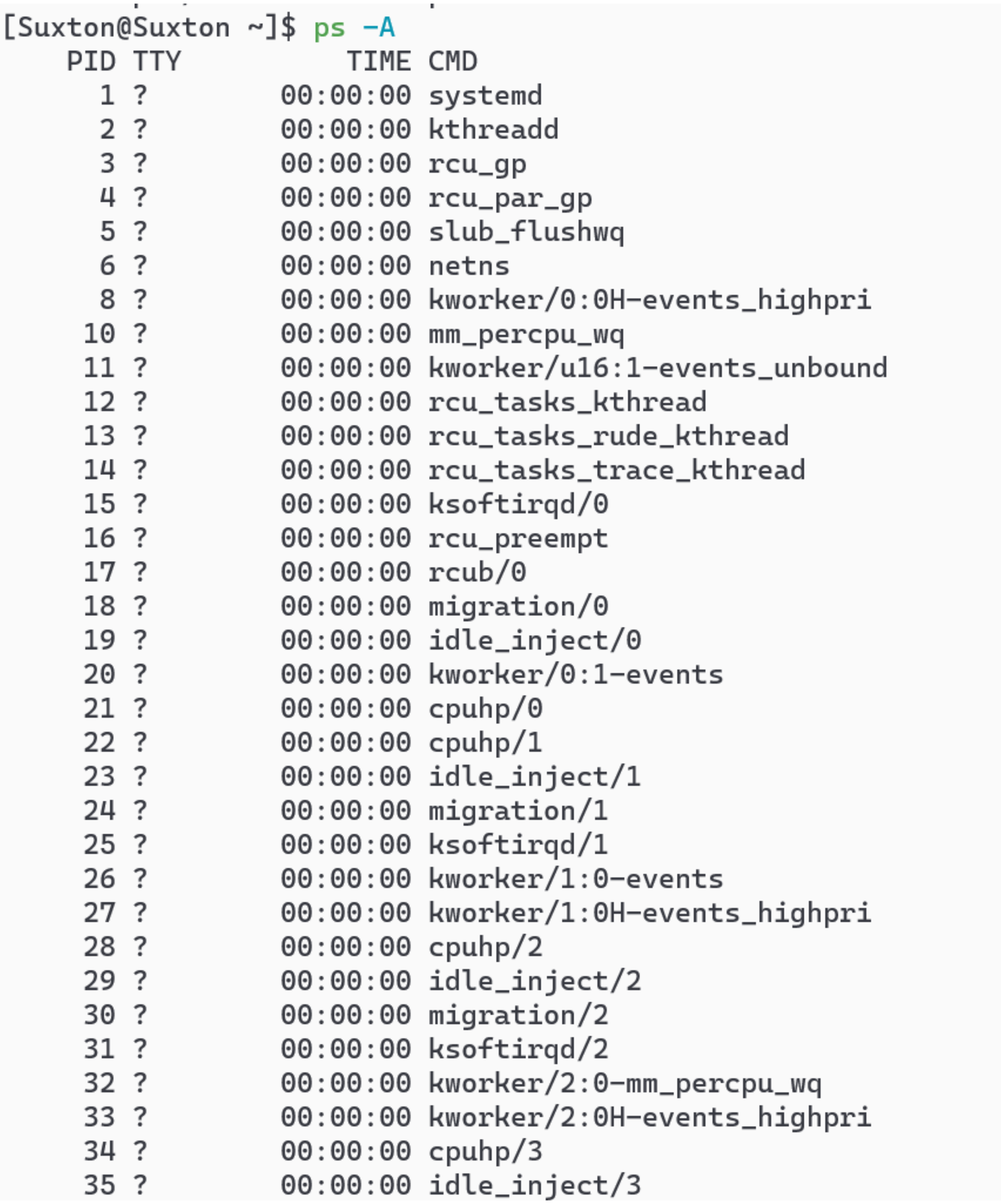
7. tar 命令：用于归档，类似于压缩文件。-cf用于创建一个档案，-xf用于释放档案中的文件。下面的截图中，我先创建了有两个文件的tar，再把两个文件删除，再解压了tar文件。



```
[Suxton@Suxton ~]$ tar -cf 1.tar 1.txt 1-copy.txt
[Suxton@Suxton ~]$ tar -xf 1.tar
[Suxton@Suxton ~]$ ls
code  documents  linux-vm-tools  1-copy.txt  1.tar  1.txt
[Suxton@Suxton ~]$ tar -xf 1.tar ./documents
tar: ./documents: Not found in archive
tar: Exiting with failure status due to previous errors
[Suxton@Suxton ~]$ tar -xf 1.tar
[Suxton@Suxton ~]$ rm 1.txt
rm: remove regular empty file '1.txt'? y
[Suxton@Suxton ~]$ rm 1-copy.txt
rm: remove regular empty file '1-copy.txt'? y
[Suxton@Suxton ~]$ ls
code  documents  linux-vm-tools  1.tar
[Suxton@Suxton ~]$ tar -xf 1.tar
[Suxton@Suxton ~]$ ls
code  documents  linux-vm-tools  1-copy.txt  1.tar  1.txt
[Suxton@Suxton ~]$ _
```

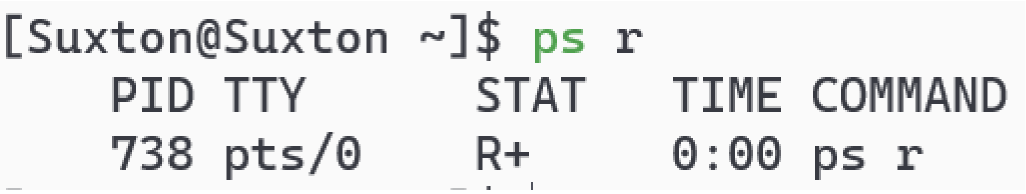
8. ps命令：用于查看运行的程序

查看全部进程



```
[Suxton@Suxton ~]$ ps -A
    PID TTY          TIME CMD
      1 ?        00:00:00 systemd
      2 ?        00:00:00 kthreadd
      3 ?        00:00:00 rcu_gp
      4 ?        00:00:00 rcu_par_gp
      5 ?        00:00:00 slub_flushwq
      6 ?        00:00:00 netns
      8 ?        00:00:00 kworker/0:0H-events_highpri
     10 ?        00:00:00 mm_percpu_wq
     11 ?        00:00:00 kworker/u16:1-events_unbound
     12 ?        00:00:00 rcu_tasks_kthread
     13 ?        00:00:00 rcu_tasks_rude_kthread
     14 ?        00:00:00 rcu_tasks_trace_kthread
     15 ?        00:00:00 ksoftirqd/0
     16 ?        00:00:00 rcu_preempt
     17 ?        00:00:00 rcub/0
     18 ?        00:00:00 migration/0
     19 ?        00:00:00 idle_inject/0
     20 ?        00:00:00 kworker/0:1-events
     21 ?        00:00:00 cpuhp/0
     22 ?        00:00:00 cpuhp/1
     23 ?        00:00:00 idle_inject/1
     24 ?        00:00:00 migration/1
     25 ?        00:00:00 ksoftirqd/1
     26 ?        00:00:00 kworker/1:0-events
     27 ?        00:00:00 kworker/1:0H-events_highpri
     28 ?        00:00:00 cpuhp/2
     29 ?        00:00:00 idle_inject/2
     30 ?        00:00:00 migration/2
     31 ?        00:00:00 ksoftirqd/2
     32 ?        00:00:00 kworker/2:0-mm_percpu_wq
     33 ?        00:00:00 kworker/2:0H-events_highpri
     34 ?        00:00:00 cpuhp/3
     35 ?        00:00:00 idle_inject/3
```

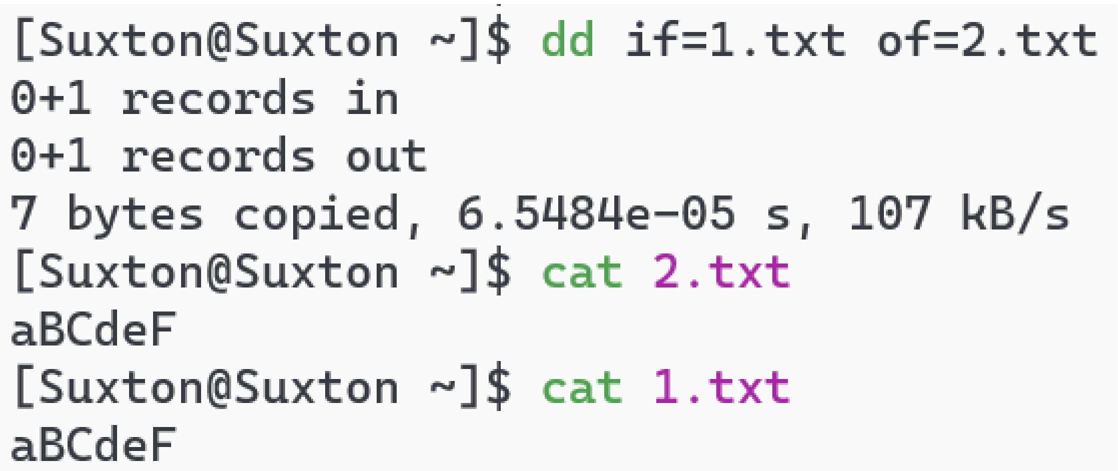
查看正在运行的进程



```
[Suxton@Suxton ~]$ ps r
    PID TTY      STAT   TIME COMMAND
    738 pts/0    R+     0:00 ps r
```

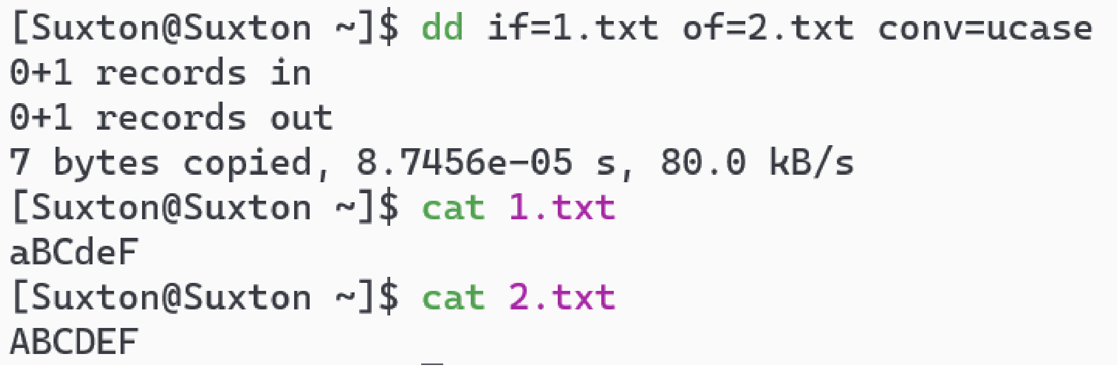
9. dd命令：将一个文件的内容拷贝到另一个文件，可以对数据进行一些处理

先建立一个文本文件，然后正常拷贝



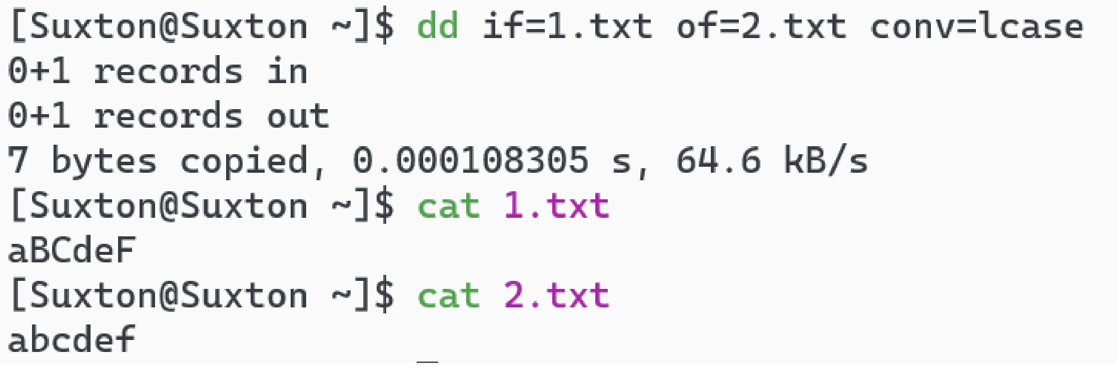
```
[Suxton@Suxton ~]$ dd if=1.txt of=2.txt
0+1 records in
0+1 records out
7 bytes copied, 6.5484e-05 s, 107 kB/s
[Suxton@Suxton ~]$ cat 2.txt
aBCdeF
[Suxton@Suxton ~]$ cat 1.txt
aBCdeF
```

拷贝的时候将小写转换为大写



```
[Suxton@Suxton ~]$ dd if=1.txt of=2.txt conv=ucase
0+1 records in
0+1 records out
7 bytes copied, 8.7456e-05 s, 80.0 kB/s
[Suxton@Suxton ~]$ cat 1.txt
aBCdeF
[Suxton@Suxton ~]$ cat 2.txt
ABCDEF
```

拷贝的时候将大写转换为小写

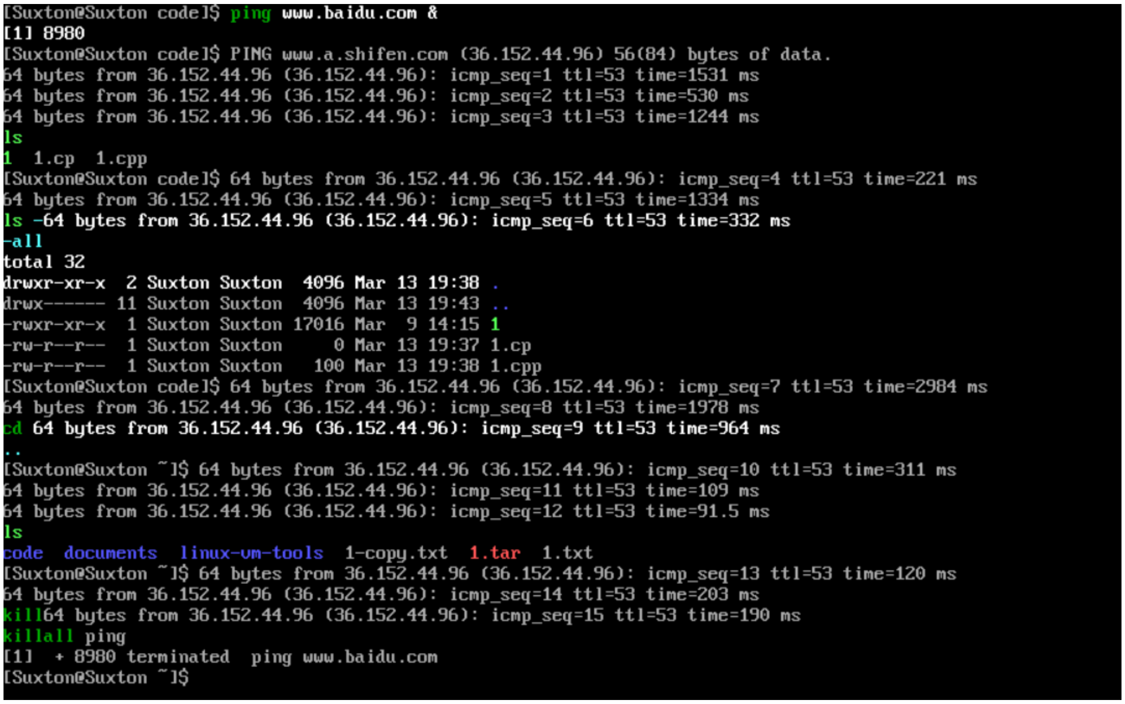


```
[Suxton@Suxton ~]$ dd if=1.txt of=2.txt conv=lcase
0+1 records in
0+1 records out
7 bytes copied, 0.000108305 s, 64.6 kB/s
[Suxton@Suxton ~]$ cat 1.txt
aBCdeF
[Suxton@Suxton ~]$ cat 2.txt
abcdef
```

3. **前后台执行**

    1. 在终端中直接输入命令就是前台执行，上面所有的截图都是前台执行的。

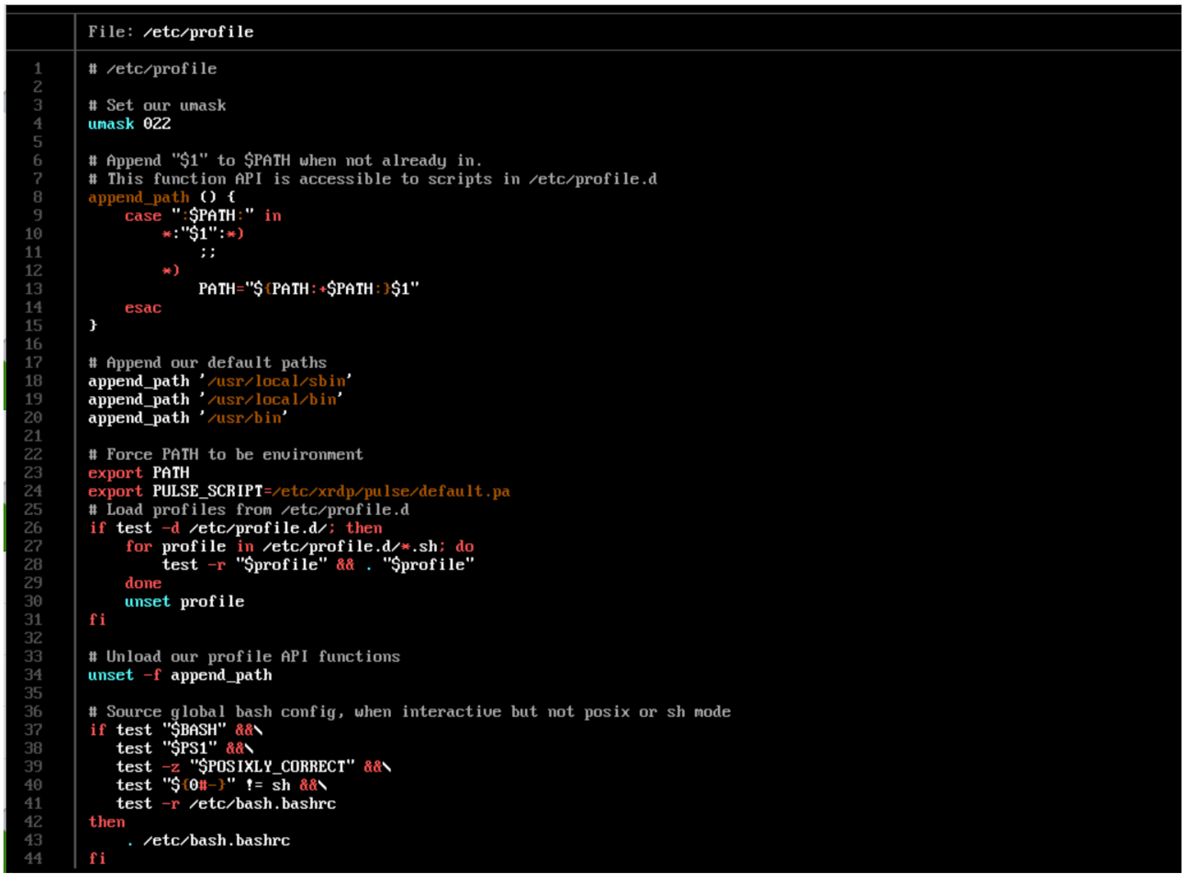
2. 在命令后面加上&就能将一个程序放在后台执行。在截图中，我在后台执行了ping命令，终端会输出ping的内容，同时我可以继续执行新的命令。直到我结束了ping程序。



```
[Suxton@Suxton code]$ ping www.baidu.com &
[1] 8980
[Suxton@Suxton code]$ PING www.a.shifen.com (36.152.44.96) 56(84) bytes of data.
64 bytes from 36.152.44.96 (36.152.44.96): icmp_seq=1 ttl=53 time=1531 ms
64 bytes from 36.152.44.96 (36.152.44.96): icmp_seq=2 ttl=53 time=530 ms
64 bytes from 36.152.44.96 (36.152.44.96): icmp_seq=3 ttl=53 time=1244 ms
ls
1  1.cp  1.cpp
[Suxton@Suxton code]$ 64 bytes from 36.152.44.96 (36.152.44.96): icmp_seq=4 ttl=53 time=221 ms
64 bytes from 36.152.44.96 (36.152.44.96): icmp_seq=5 ttl=53 time=1334 ms
ls -64 bytes from 36.152.44.96 (36.152.44.96): icmp_seq=6 ttl=53 time=332 ms
-all
total 32
drwxr-xr-x  2 Suxton Suxton  4096 Mar 13 19:38 .
drwx------ 11 Suxton Suxton  4096 Mar 13 19:43 ..
-rwxr-xr-x  1 Suxton Suxton 17016 Mar  9 14:15 1
-rw-r--r--  1 Suxton Suxton     0 Mar 13 19:37 1.cp
-rw-r--r--  1 Suxton Suxton   100 Mar 13 19:38 1.cpp
[Suxton@Suxton code]$ 64 bytes from 36.152.44.96 (36.152.44.96): icmp_seq=7 ttl=53 time=2984 ms
64 bytes from 36.152.44.96 (36.152.44.96): icmp_seq=8 ttl=53 time=1978 ms
cd 64 bytes from 36.152.44.96 (36.152.44.96): icmp_seq=9 ttl=53 time=964 ms
..
[Suxton@Suxton ~]$ 64 bytes from 36.152.44.96 (36.152.44.96): icmp_seq=10 ttl=53 time=311 ms
64 bytes from 36.152.44.96 (36.152.44.96): icmp_seq=11 ttl=53 time=109 ms
64 bytes from 36.152.44.96 (36.152.44.96): icmp_seq=12 ttl=53 time=91.5 ms
ls
code  documents  linux-vm-tools  1-copy.txt  1.tar  1.txt
[Suxton@Suxton ~]$ 64 bytes from 36.152.44.96 (36.152.44.96): icmp_seq=13 ttl=53 time=120 ms
64 bytes from 36.152.44.96 (36.152.44.96): icmp_seq=14 ttl=53 time=203 ms
kill64 bytes from 36.152.44.96 (36.152.44.96): icmp_seq=15 ttl=53 time=190 ms
killall ping
[1]  + 8980 terminated  ping www.baidu.com
[Suxton@Suxton ~]$
```

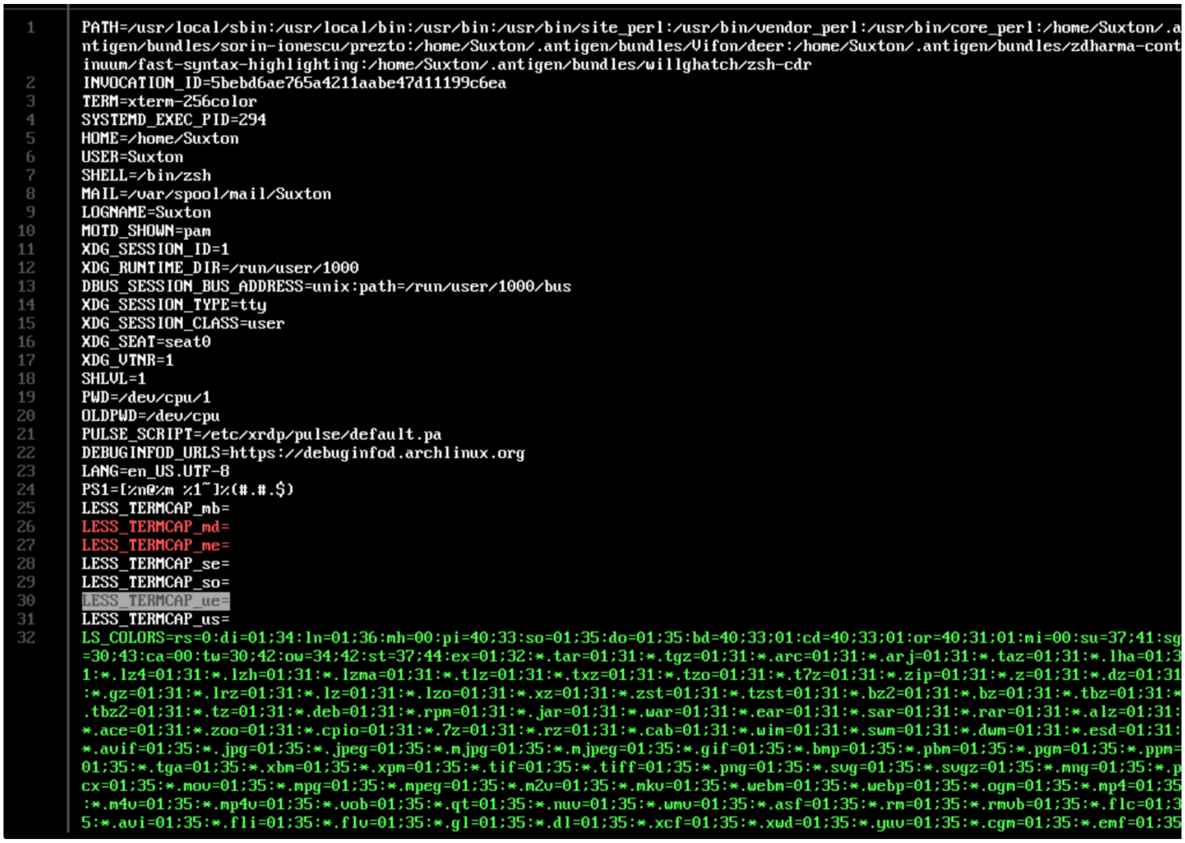
4. **环境配置文件**

Linux的环境配置文件是/etc/profile中，我使用bat（cat的加强版）工具查看。



```
File: /etc/profile

# /etc/profile

# Set our umask
umask 022

# Append "$1" to $PATH when not already in.
# This function API is accessible to scripts in /etc/profile.d
append_path () {
    case ":$PATH:" in
        *:"$1":*)
            ;;
        *)
            PATH="${PATH:+$PATH:}$1"
    esac
}

# Append our default paths
append_path '/usr/local/sbin'
append_path '/usr/local/bin'
append_path '/usr/bin'

# Force PATH to be environment
export PATH
export PULSE_SCRIPT=/etc/xrdp/pulse/default.pa
# Load profiles from /etc/profile.d
if test -d /etc/profile.d/; then
    for profile in /etc/profile.d/*.sh; do
        test -r "$profile" && . "$profile"
    done
    unset profile
fi

# Unload our profile API functions
unset -f append_path

# Source global bash config, when interactive but not posix or sh mode
if test "$BASH" &&\
   test "$PS1" &&\
   test -z "$POSIXLY_CORRECT" &&\
   test "${0#-}" != sh &&\
   test -r /etc/bash.bashrc
then
    . /etc/bash.bashrc
fi
```

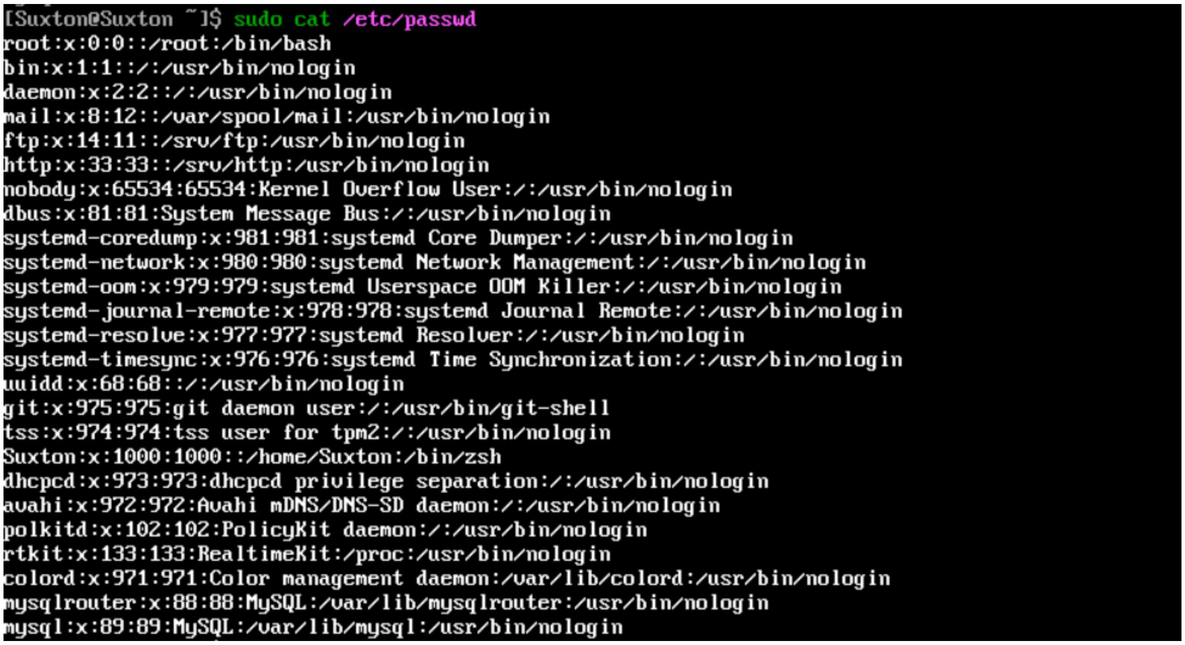
输入env就能查看环境变量，下面使用管道传入bat查看。



```
PATH=/usr/local/sbin:/usr/local/bin:/usr/bin:/usr/bin/site_perl:/usr/bin/vendor_perl:/usr/bin/core_perl:/home/Suxton/.a
ntigen/bundles/sorin-ionescu/prezto:/home/Suxton/.antigen/bundles/Vifon/deer:/home/Suxton/.antigen/bundles/zdharma-cont
inuum/fast-syntax-highlighting:/home/Suxton/.antigen/bundles/willghatch/zsh-cdr
INVOCATION_ID=5bebd6ae765a4211aabe47d11199c6ea
TERM=xterm-256color
SYSTEMD_EXEC_PID=294
HOME=/home/Suxton
USER=Suxton
SHELL=/bin/zsh
MAIL=/var/spool/mail/Suxton
LOGNAME=Suxton
MOTD_SHOWN=pam
XDG_SESSION_ID=1
XDG_RUNTIME_DIR=/run/user/1000
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
XDG_SESSION_TYPE=tty
XDG_SESSION_CLASS=user
XDG_SEAT=seat0
XDG_VTNR=1
SHLVL=1
PWD=/dev/cpu/1
OLDPWD=/dev/cpu
PULSE_SCRIPT=/etc/xrdp/pulse/default.pa
DEBUGINFOD_URLS=https://debuginfod.archlinux.org
LANG=en_US.UTF-8
PS1=[%n@%m %1~]%(#.#.$)
LESS_TERMCAP_mb=
LESS_TERMCAP_md=
LESS_TERMCAP_me=
LESS_TERMCAP_se=
LESS_TERMCAP_so=
LESS_TERMCAP_ue=
LESS_TERMCAP_us=
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=40;31;01:mi=00:su=37;41:sg
=30;43:ca=00:tw=30;42:ow=34;42:st=37;44:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;3
1:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;31:*.t7z=01;31:*.zip=01;31:*.z=01;31:*.dz=01;31
:*.gz=01;31:*.lrz=01;31:*.lz=01;31:*.lzo=01;31:*.xz=01;31:*.zst=01;31:*.tzst=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*
.tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;31:*.sar=01;31:*.rar=01;31:*.alz=01;31:
*.ace=01;31:*.zoo=01;31:*.cpio=01;31:*.7z=01;31:*.rz=01;31:*.cab=01;31:*.wim=01;31:*.swm=01;31:*.dwm=01;31:*.esd=01;31:
*.avif=01;35:*.jpg=01;35:*.jpeg=01;35:*.mjpg=01;35:*.mjpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=01;35:*.pgm=01;35:*.ppm=
01;35:*.tga=01;35:*.xbm=01;35:*.xpm=01;35:*.tif=01;35:*.tiff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=01;35:*.p
cx=01;35:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35:*.m2v=01;35:*.mkv=01;35:*.webm=01;35:*.webp=01;35:*.ogm=01;35:*.mp4=01;35
:*.m4v=01;35:*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35:*.asf=01;35:*.rm=01;35:*.rmvb=01;35:*.flc=01;3
5:*.avi=01;35:*.fli=01;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xcf=01;35:*.xwd=01;35:*.yuv=01;35:*.cgm=01;35:*.emf=01;35
```

5. 用户信息文件

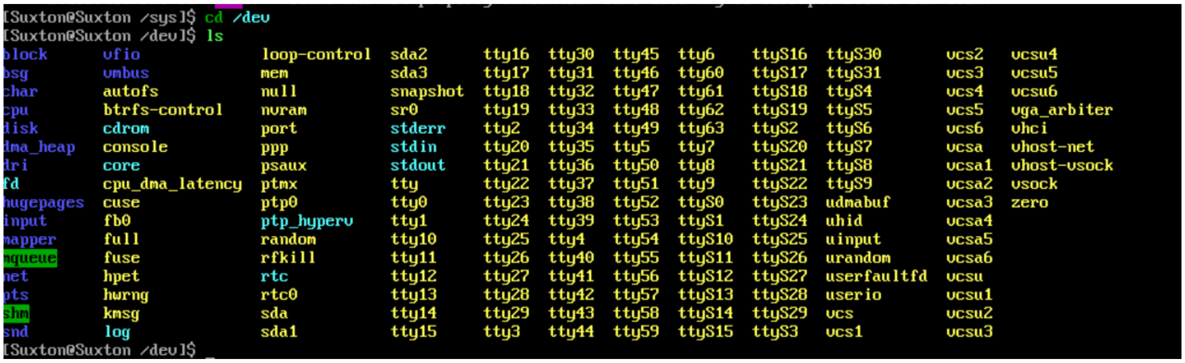
用户信息存在/etc/passwd，使用cat查看



```
[Suxton@Suxton ~]$ sudo cat /etc/passwd
root:x:0:0::/root:/bin/bash
bin:x:1:1::/:/usr/bin/nologin
daemon:x:2:2::/:/usr/bin/nologin
mail:x:8:12::/var/spool/mail:/usr/bin/nologin
ftp:x:14:11::/srv/ftp:/usr/bin/nologin
http:x:33:33::/srv/http:/usr/bin/nologin
nobody:x:65534:65534:Kernel Overflow User:/:/usr/bin/nologin
dbus:x:81:81:System Message Bus:/:/usr/bin/nologin
systemd-coredump:x:981:981:systemd Core Dumper:/:/usr/bin/nologin
systemd-network:x:980:980:systemd Network Management:/:/usr/bin/nologin
systemd-oom:x:979:979:systemd Userspace OOM Killer:/:/usr/bin/nologin
systemd-journal-remote:x:978:978:systemd Journal Remote:/:/usr/bin/nologin
systemd-resolve:x:977:977:systemd Resolver:/:/usr/bin/nologin
systemd-timesync:x:976:976:systemd Time Synchronization:/:/usr/bin/nologin
uuidd:x:68:68::/:/usr/bin/nologin
git:x:975:975:git daemon user:/:/usr/bin/git-shell
tss:x:974:974:tss user for tpm2:/:/usr/bin/nologin
Suxton:x:1000:1000::/home/Suxton:/bin/zsh
dhcpcd:x:973:973:dhcpcd privilege separation:/:/usr/bin/nologin
avahi:x:972:972:Avahi mDNS/DNS-SD daemon:/:/usr/bin/nologin
polkitd:x:102:102:PolicyKit daemon:/:/usr/bin/nologin
rtkit:x:133:133:RealtimeKit:/proc:/usr/bin/nologin
colord:x:971:971:Color management daemon:/var/lib/colord:/usr/bin/nologin
mysqlrouter:x:88:88:MySQL:/var/lib/mysqlrouter:/usr/bin/nologin
mysql:x:89:89:MySQL:/var/lib/mysql:/usr/bin/nologin
```

6. **设备加载信息**

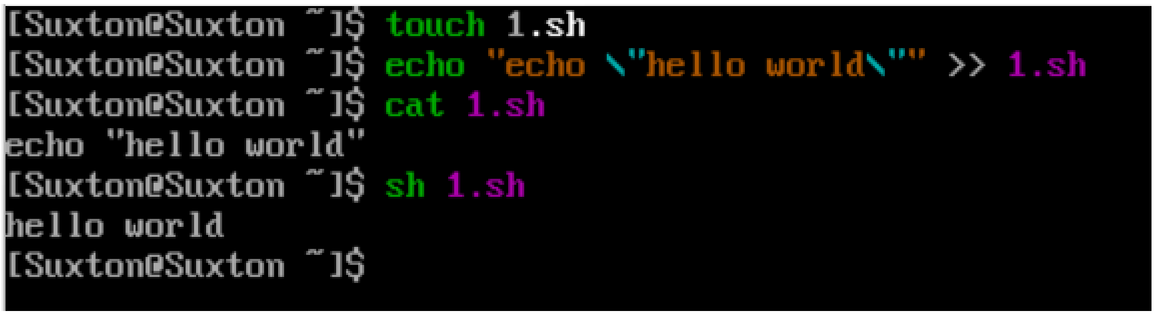
Linux下所有的设备都会出现在/dev中，下面使用ls查看。



```
[Suxton@Suxton /sys]$ cd /dev
[Suxton@Suxton /dev]$ ls
block      vfio             loop-control  sda2      tty16  tty30  tty45  tty6   ttyS16  ttyS30       vcs2   vcsu4
bsg        vmbus            mem           sda3      tty17  tty31  tty46  tty60  ttyS17  ttyS31       vcs3   vcsu5
char       autofs           null          snapshot  tty18  tty32  tty47  tty61  ttyS18  ttyS4        vcs4   vcsu6
cpu        btrfs-control    nvram         sr0       tty19  tty33  tty48  tty62  ttyS19  ttyS5        vcs5   vga_arbiter
disk       cdrom            port          stderr    tty2   tty34  tty49  tty63  ttyS2   ttyS6        vcs6   vhci
dma_heap   console          ppp           stdin     tty20  tty35  tty5   tty7   ttyS20  ttyS7        vcsa   vhost-net
dri        core             psaux         stdout    tty21  tty36  tty50  tty8   ttyS21  ttyS8        vcsa1  vhost-vsock
fd         cpu_dma_latency  ptmx          tty       tty22  tty37  tty51  tty9   ttyS22  ttyS9        vcsa2  vsock
hugepages  cuse             ptp0          tty0      tty23  tty38  tty52  ttyS0  ttyS23  udmabuf      vcsa3  zero
input      fb0              ptp_hyperv    tty1      tty24  tty39  tty53  ttyS1  ttyS24  uhid         vcsa4
mapper     full             random        tty10     tty25  tty4   tty54  ttyS10 ttyS25  uinput       vcsa5
mqueue     fuse             rfkill        tty11     tty26  tty40  tty55  ttyS11 ttyS26  urandom      vcsa6
net        hpet             rtc           tty12     tty27  tty41  tty56  ttyS12 ttyS27  userfaultfd  vcsu
pts        hwrng            rtc0          tty13     tty28  tty42  tty57  ttyS13 ttyS28  userio       vcsu1
shm        kmsg             sda           tty14     tty29  tty43  tty58  ttyS14 ttyS29  vcs          vcsu2
snd        log              sda1          tty15     tty3   tty44  tty59  ttyS15 ttyS3   vcs1         vcsu3
[Suxton@Suxton /dev]$ _
```

7. **系统启动脚本文件**

我自己写了个脚本，然后执行了一下

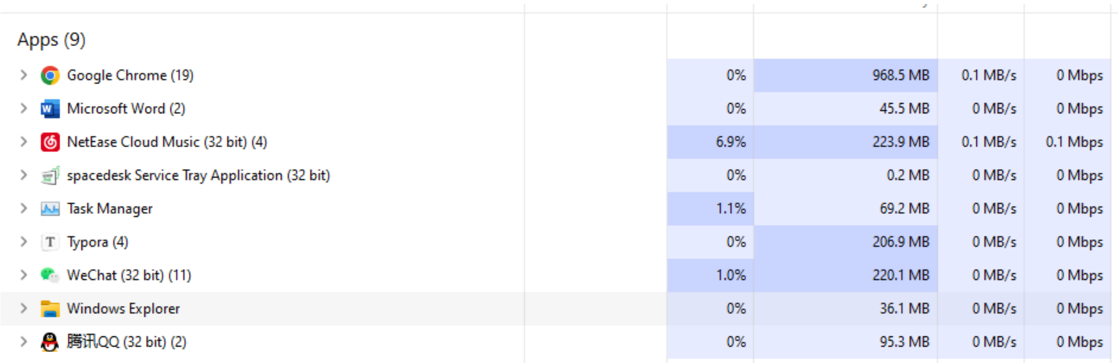


```
[Suxton@Suxton ~]$ touch 1.sh
[Suxton@Suxton ~]$ echo "echo \"hello world\"" >> 1.sh
[Suxton@Suxton ~]$ cat 1.sh
echo "hello world"
[Suxton@Suxton ~]$ sh 1.sh
hello world
[Suxton@Suxton ~]$
```

Windows部分

1. **了解进程和服务**

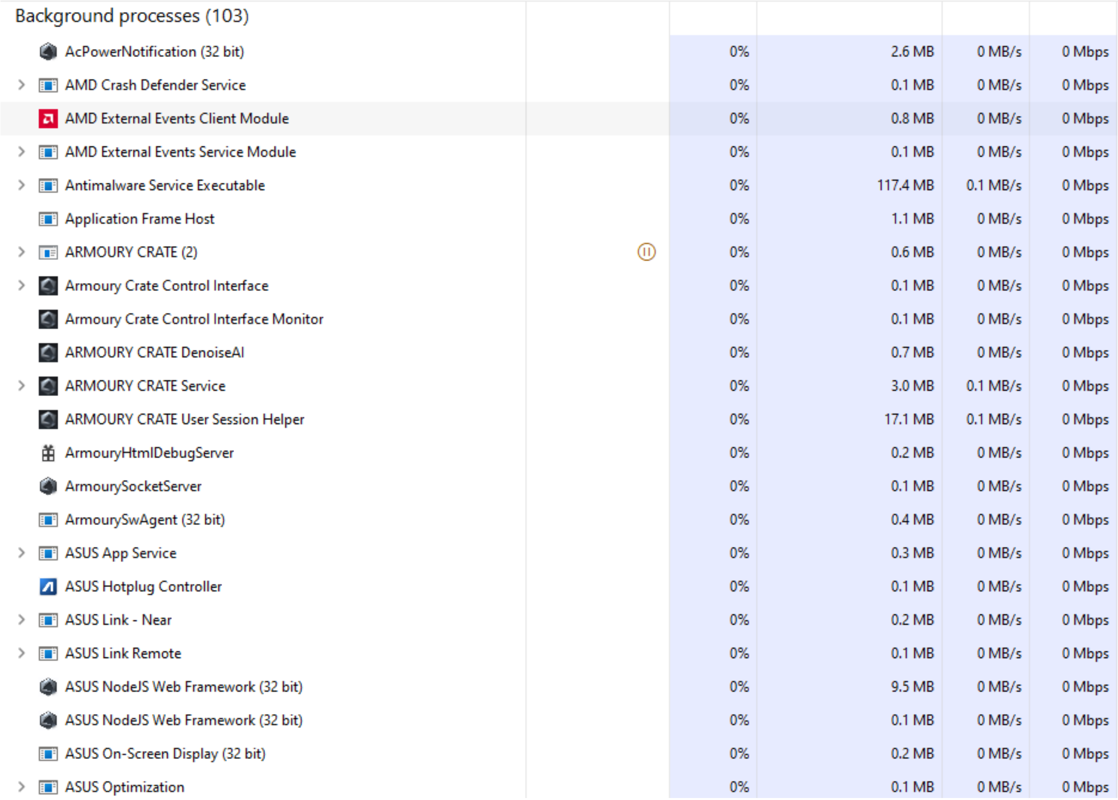
    1. 用户的进程



Apps (9)					
Google Chrome (19)		0%	968.5 MB	0.1 MB/s	0 Mbps
Microsoft Word (2)		0%	45.5 MB	0 MB/s	0 Mbps
NetEase Cloud Music (32 bit) (4)		6.9%	223.9 MB	0.1 MB/s	0.1 Mbps
spacedesk Service Tray Application (32 bit)		0%	0.2 MB	0 MB/s	0 Mbps
Task Manager		1.1%	69.2 MB	0 MB/s	0 Mbps
Typora (4)		0%	206.9 MB	0 MB/s	0 Mbps
WeChat (32 bit) (11)		1.0%	220.1 MB	0 MB/s	0 Mbps
Windows Explorer		0%	36.1 MB	0 MB/s	0 Mbps
腾讯QQ (32 bit) (2)		0%	95.3 MB	0 MB/s	0 Mbps

    2. 系统的进程（后台运行）

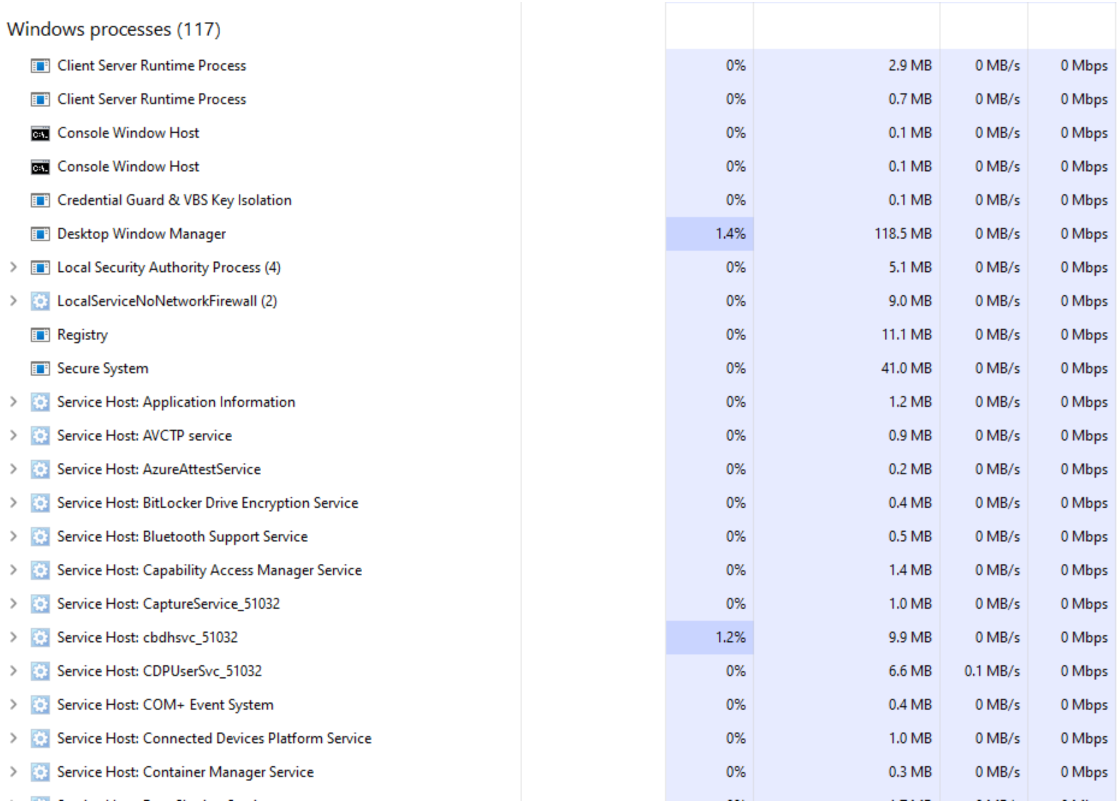
        这里面有华硕还有AMD还有英伟达的驱动程序进程，都是由系统执行的。



Background processes (103)					
AcPowerNotification (32 bit)		0%	2.6 MB	0 MB/s	0 Mbps
AMD Crash Defender Service		0%	0.1 MB	0 MB/s	0 Mbps
AMD External Events Client Module		0%	0.8 MB	0 MB/s	0 Mbps
AMD External Events Service Module		0%	0.1 MB	0 MB/s	0 Mbps
Antimalware Service Executable		0%	117.4 MB	0.1 MB/s	0 Mbps
Application Frame Host		0%	1.1 MB	0 MB/s	0 Mbps
ARMOURY CRATE (2)	⏸	0%	0.6 MB	0 MB/s	0 Mbps
Armoury Crate Control Interface		0%	0.1 MB	0 MB/s	0 Mbps
Armoury Crate Control Interface Monitor		0%	0.1 MB	0 MB/s	0 Mbps
ARMOURY CRATE DenoiseAI		0%	0.7 MB	0 MB/s	0 Mbps
ARMOURY CRATE Service		0%	3.0 MB	0.1 MB/s	0 Mbps
ARMOURY CRATE User Session Helper		0%	17.1 MB	0.1 MB/s	0 Mbps
ArmouryHtmlDebugServer		0%	0.2 MB	0 MB/s	0 Mbps
ArmourySocketServer		0%	0.1 MB	0 MB/s	0 Mbps
ArmourySwAgent (32 bit)		0%	0.4 MB	0 MB/s	0 Mbps
ASUS App Service		0%	0.3 MB	0 MB/s	0 Mbps
ASUS Hotplug Controller		0%	0.1 MB	0 MB/s	0 Mbps
ASUS Link - Near		0%	0.2 MB	0 MB/s	0 Mbps
ASUS Link Remote		0%	0.1 MB	0 MB/s	0 Mbps
ASUS NodeJS Web Framework (32 bit)		0%	9.5 MB	0 MB/s	0 Mbps
ASUS NodeJS Web Framework (32 bit)		0%	0.1 MB	0 MB/s	0 Mbps
ASUS On-Screen Display (32 bit)		0%	0.2 MB	0 MB/s	0 Mbps
ASUS Optimization		0%	0.1 MB	0 MB/s	0 Mbps

    3. 系统的服务

        齿轮图标的都是服务进程，其他的是普通进程，都是系统提供的服务。



Windows processes (117)

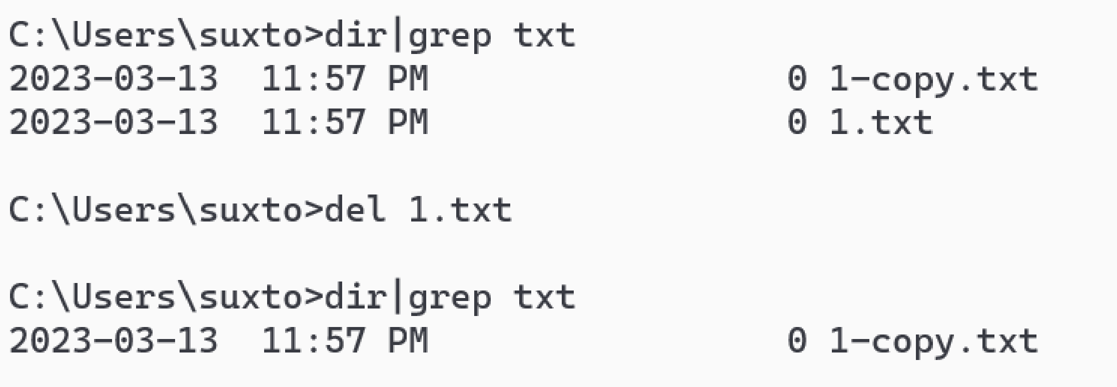
Client Server Runtime Process	0%	2.9 MB	0 MB/s	0 Mbps
Client Server Runtime Process	0%	0.7 MB	0 MB/s	0 Mbps
Console Window Host	0%	0.1 MB	0 MB/s	0 Mbps
Console Window Host	0%	0.1 MB	0 MB/s	0 Mbps
Credential Guard & VBS Key Isolation	0%	0.1 MB	0 MB/s	0 Mbps
Desktop Window Manager	1.4%	118.5 MB	0 MB/s	0 Mbps
Local Security Authority Process (4)	0%	5.1 MB	0 MB/s	0 Mbps
LocalServiceNoNetworkFirewall (2)	0%	9.0 MB	0 MB/s	0 Mbps
Registry	0%	11.1 MB	0 MB/s	0 Mbps
Secure System	0%	41.0 MB	0 MB/s	0 Mbps
Service Host: Application Information	0%	1.2 MB	0 MB/s	0 Mbps
Service Host: AVCTP service	0%	0.9 MB	0 MB/s	0 Mbps
Service Host: AzureAttestService	0%	0.2 MB	0 MB/s	0 Mbps
Service Host: BitLocker Drive Encryption Service	0%	0.4 MB	0 MB/s	0 Mbps
Service Host: Bluetooth Support Service	0%	0.5 MB	0 MB/s	0 Mbps
Service Host: Capability Access Manager Service	0%	1.4 MB	0 MB/s	0 Mbps
Service Host: CaptureService_51032	0%	1.0 MB	0 MB/s	0 Mbps
Service Host: cbdhsvc_51032	1.2%	9.9 MB	0 MB/s	0 Mbps
Service Host: CDPUserSvc_51032	0%	6.6 MB	0.1 MB/s	0 Mbps
Service Host: COM+ Event System	0%	0.4 MB	0 MB/s	0 Mbps
Service Host: Connected Devices Platform Service	0%	1.0 MB	0 MB/s	0 Mbps
Service Host: Container Manager Service	0%	0.3 MB	0 MB/s	0 Mbps

2. **常用命令**

   1. copy 命令：我创建了一个1.txt文件，再用copy命令复制了一遍。可能是我用的Windows11，很多命令被改进了，这个命令和Linux没啥区别。

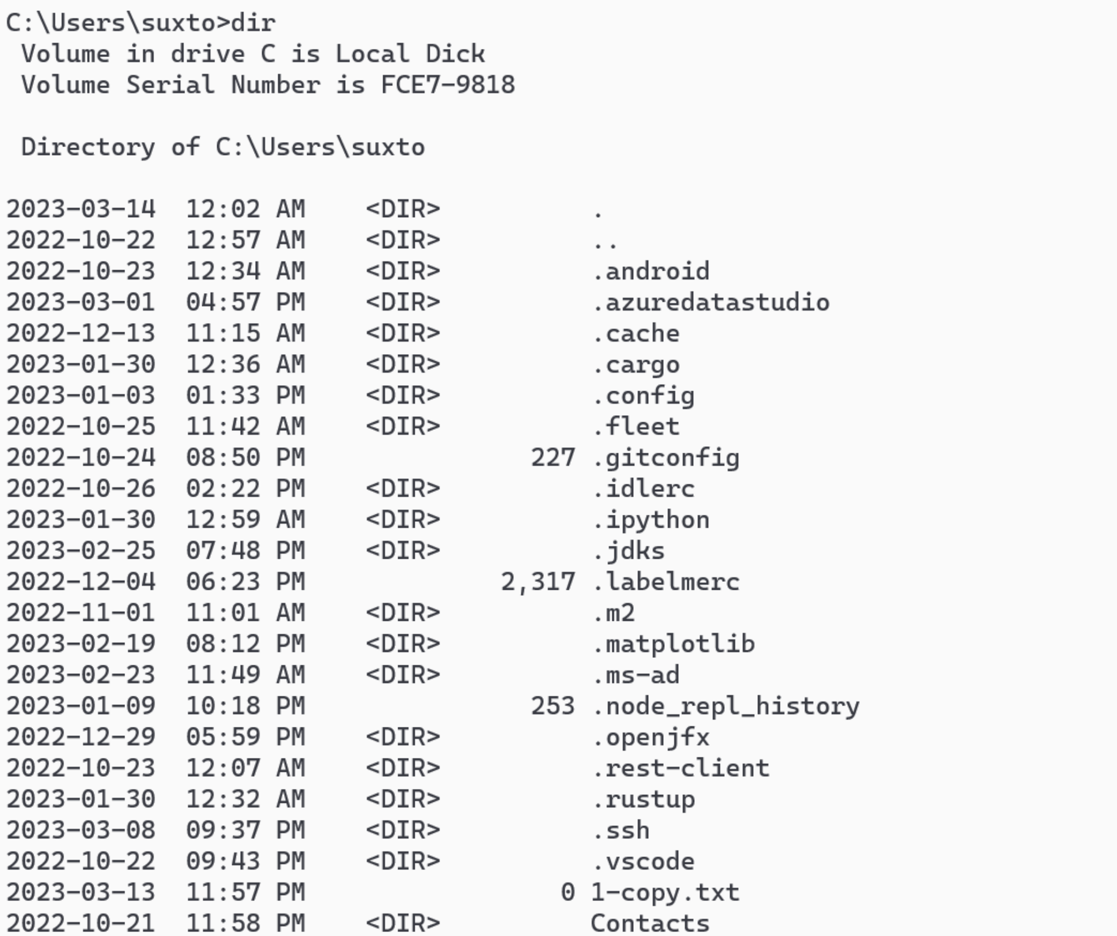
```
C:\Users\suxto>dir|grep txt

C:\Users\suxto>touch 1.txt

C:\Users\suxto>dir|grep txt
2023-03-13  11:57 PM                 0 1.txt

C:\Users\suxto>copy 1.txt 1-copy.txt
        1 file(s) copied.

C:\Users\suxto>dir|grep txt
2023-03-13  11:57 PM                 0 1-copy.txt
2023-03-13  11:57 PM                 0 1.txt
```

   2. del 命令：我把刚刚创建的1.txt文件删除了，感觉和Linux没啥区别。



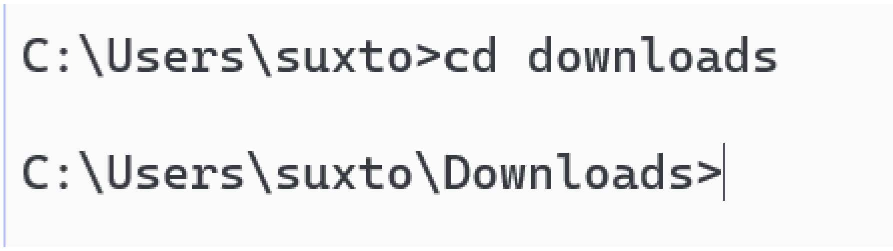
```
C:\Users\suxto>dir|grep txt
2023-03-13  11:57 PM                 0 1-copy.txt
2023-03-13  11:57 PM                 0 1.txt

C:\Users\suxto>del 1.txt

C:\Users\suxto>dir|grep txt
2023-03-13  11:57 PM                 0 1-copy.txt
```

3. dir 命令：显示当前的目录下所有文件和目录，相当于ls -a



```
C:\Users\suxto>dir
 Volume in drive C is Local Dick
 Volume Serial Number is FCE7-9818

 Directory of C:\Users\suxto

2023-03-14  12:02 AM    <DIR>          .
2022-10-22  12:57 AM    <DIR>          ..
2022-10-23  12:34 AM    <DIR>          .android
2023-03-01  04:57 PM    <DIR>          .azuredatastudio
2022-12-13  11:15 AM    <DIR>          .cache
2023-01-30  12:36 AM    <DIR>          .cargo
2023-01-03  01:33 PM    <DIR>          .config
2022-10-25  11:42 AM    <DIR>          .fleet
2022-10-24  08:50 PM               227 .gitconfig
2022-10-26  02:22 PM    <DIR>          .idlerc
2023-01-30  12:59 AM    <DIR>          .ipython
2023-02-25  07:48 PM    <DIR>          .jdks
2022-12-04  06:23 PM             2,317 .labelmerc
2022-11-01  11:01 AM    <DIR>          .m2
2023-02-19  08:12 PM    <DIR>          .matplotlib
2023-02-23  11:49 AM    <DIR>          .ms-ad
2023-01-09  10:18 PM               253 .node_repl_history
2022-12-29  05:59 PM    <DIR>          .openjfx
2022-10-23  12:07 AM    <DIR>          .rest-client
2023-01-30  12:32 AM    <DIR>          .rustup
2023-03-08  09:37 PM    <DIR>          .ssh
2022-10-22  09:43 PM    <DIR>          .vscode
2023-03-13  11:57 PM                 0 1-copy.txt
2022-10-21  11:58 PM    <DIR>          Contacts
```

4. cd 命令：选择一个目录，和Linux差不多



```
C:\Users\suxto>cd downloads

C:\Users\suxto\Downloads>
```

差异

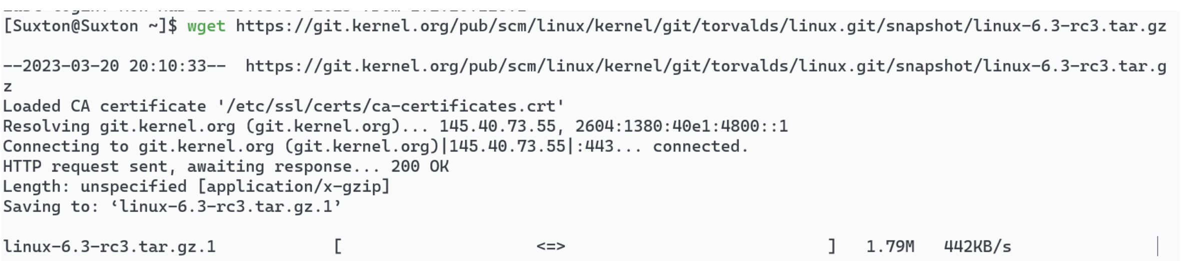
可能是我使用的是最新的Windows11，Windows的终端饱受诟病，所有可能有优化。我现在发现Linux很多命令在Windows下也可以使用，比如rm，ls，touch等。所有，Windows下的命令我并没有发现和Linux有很大的区别。

不过在Windows下不区分大小写，而Linux严格区分大小写。

编译内核

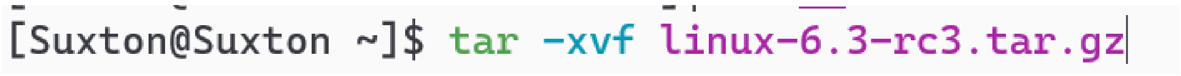
为了方便复制粘贴，我接下来都使用ssh连接虚拟机中的Linux。

1. 下载内核，使用wget命令

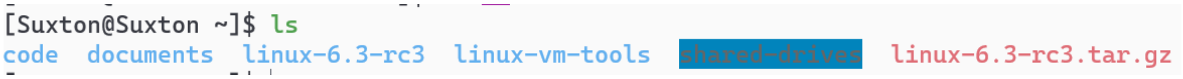


```
[Suxton@Suxton ~]$ wget https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/snapshot/linux-6.3-rc3.tar.gz

--2023-03-20 20:10:33--  https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/snapshot/linux-6.3-rc3.tar.g
z
Loaded CA certificate '/etc/ssl/certs/ca-certificates.crt'
Resolving git.kernel.org (git.kernel.org)... 145.40.73.55, 2604:1380:40e1:4800::1
Connecting to git.kernel.org (git.kernel.org)|145.40.73.55|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [application/x-gzip]
Saving to: ‘linux-6.3-rc3.tar.gz.1’

linux-6.3-rc3.tar.gz.1          [                    <=>                           ]   1.79M   442KB/s
```

2. 解压包

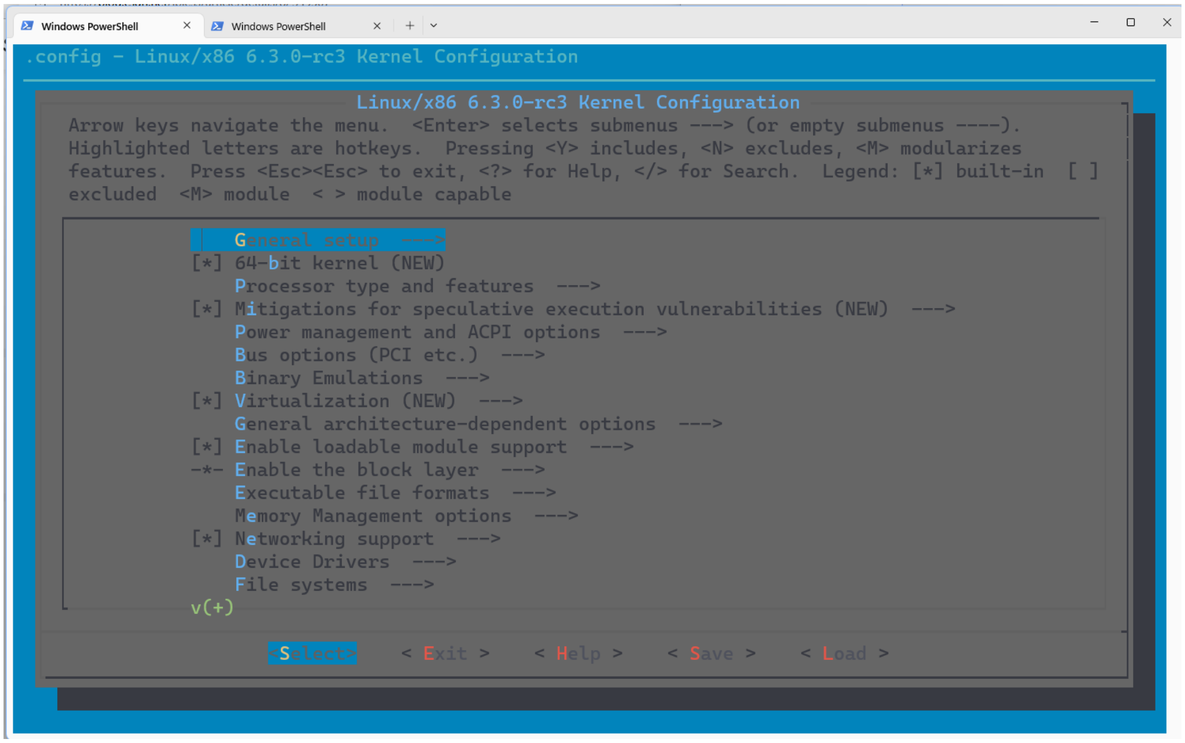


```
[Suxton@Suxton ~]$ tar -xvf linux-6.3-rc3.tar.gz
```



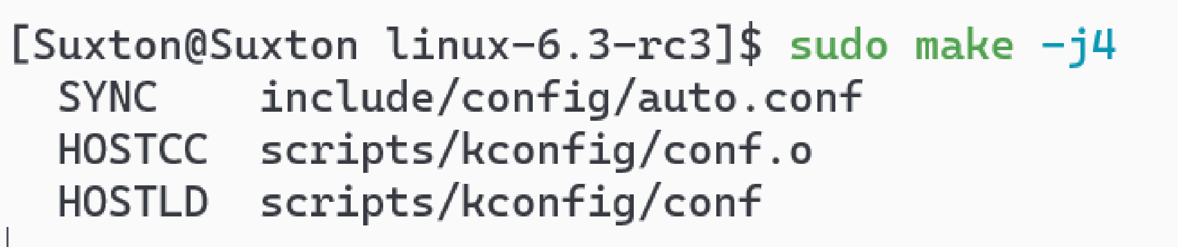
```
[Suxton@Suxton ~]$ ls
code  documents  linux-6.3-rc3  linux-vm-tools  shared-drives  linux-6.3-rc3.tar.gz
```

3. 进入文件夹，使用`make menuconfig`启动图形化界面，并配置内核

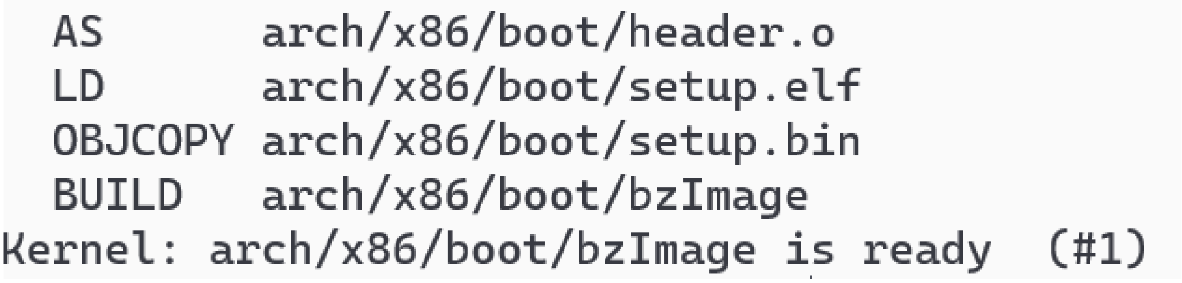


```
.config - Linux/x86 6.3.0-rc3 Kernel Configuration
                    Linux/x86 6.3.0-rc3 Kernel Configuration
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus ----).
  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M> modularizes
  features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.  Legend: [*] built-in  [ ]
  excluded  <M> module  < > module capable

                General setup  --->
            [*] 64-bit kernel (NEW)
                Processor type and features  --->
            [*] Mitigations for speculative execution vulnerabilities (NEW)  --->
                Power management and ACPI options  --->
                Bus options (PCI etc.)  --->
                Binary Emulations  --->
            [*] Virtualization (NEW)  --->
                General architecture-dependent options  --->
            [*] Enable loadable module support  --->
            -*- Enable the block layer  --->
                Executable file formats  --->
                Memory Management options  --->
            [*] Networking support  --->
                Device Drivers  --->
                File systems  --->
            v(+)

                <Select>    < Exit >    < Help >    < Save >    < Load >
```

4. 开始编译

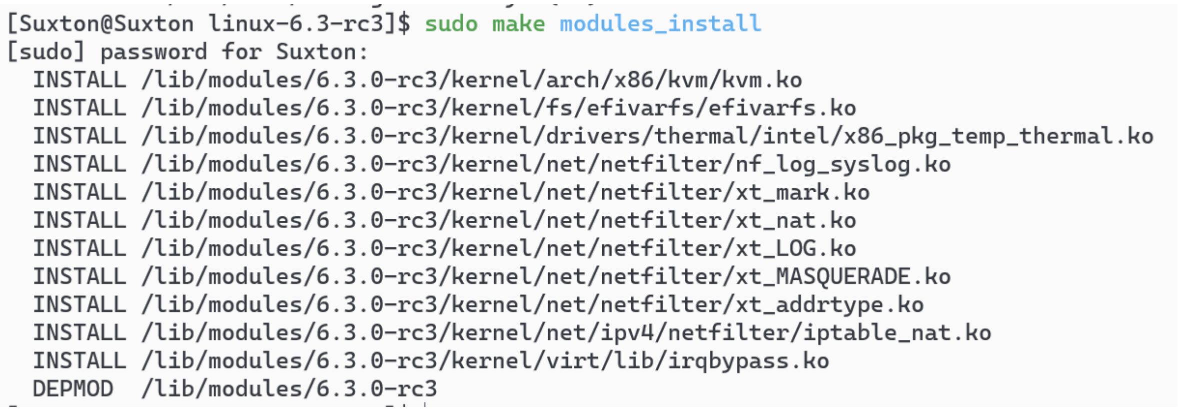


```
[Suxton@Suxton linux-6.3-rc3]$ sudo make -j4
  SYNC    include/config/auto.conf
  HOSTCC  scripts/kconfig/conf.o
  HOSTLD  scripts/kconfig/conf
```

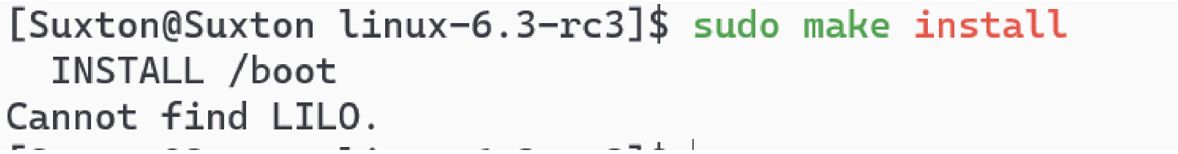


```
  AS      arch/x86/boot/header.o
  LD      arch/x86/boot/setup.elf
  OBJCOPY arch/x86/boot/setup.bin
  BUILD   arch/x86/boot/bzImage
Kernel: arch/x86/boot/bzImage is ready  (#1)
```

5. 安装内核



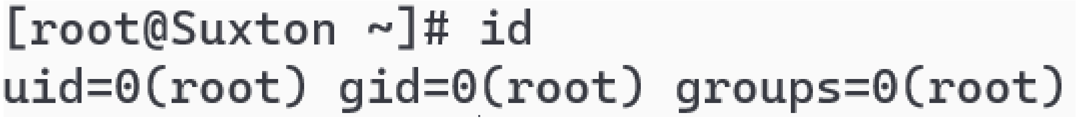
```
[Suxton@Suxton linux-6.3-rc3]$ sudo make modules_install
[sudo] password for Suxton:
  INSTALL /lib/modules/6.3.0-rc3/kernel/arch/x86/kvm/kvm.ko
  INSTALL /lib/modules/6.3.0-rc3/kernel/fs/efivarfs/efivarfs.ko
  INSTALL /lib/modules/6.3.0-rc3/kernel/drivers/thermal/intel/x86_pkg_temp_thermal.ko
  INSTALL /lib/modules/6.3.0-rc3/kernel/net/netfilter/nf_log_syslog.ko
  INSTALL /lib/modules/6.3.0-rc3/kernel/net/netfilter/xt_mark.ko
  INSTALL /lib/modules/6.3.0-rc3/kernel/net/netfilter/xt_nat.ko
  INSTALL /lib/modules/6.3.0-rc3/kernel/net/netfilter/xt_LOG.ko
  INSTALL /lib/modules/6.3.0-rc3/kernel/net/netfilter/xt_MASQUERADE.ko
  INSTALL /lib/modules/6.3.0-rc3/kernel/net/netfilter/xt_addrtype.ko
  INSTALL /lib/modules/6.3.0-rc3/kernel/net/ipv4/netfilter/iptable_nat.ko
  INSTALL /lib/modules/6.3.0-rc3/kernel/virt/lib/irqbypass.ko
  DEPMOD  /lib/modules/6.3.0-rc3
```



```
[Suxton@Suxton linux-6.3-rc3]$ sudo make install
  INSTALL /boot
Cannot find LILO.
```

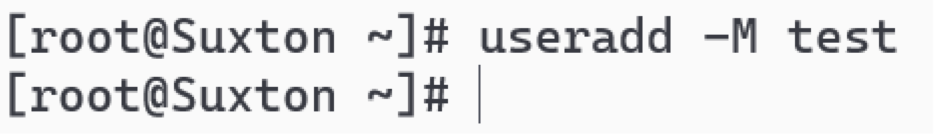
改变用户ID

1. 使用root用户登录

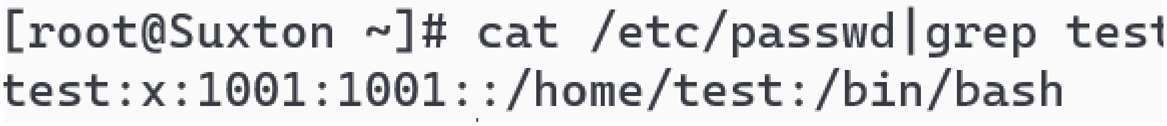


```
[root@Suxton ~]# id
uid=0(root) gid=0(root) groups=0(root)
```

2. 创建一个测试用户（不创建用户目录），通过查看passwd文件，得到id为1001

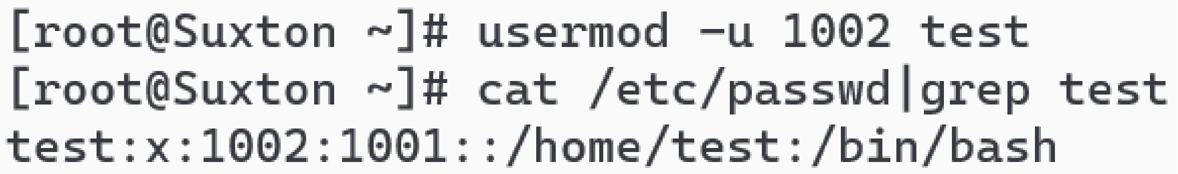


```
[root@Suxton ~]# useradd -M test
[root@Suxton ~]#
```



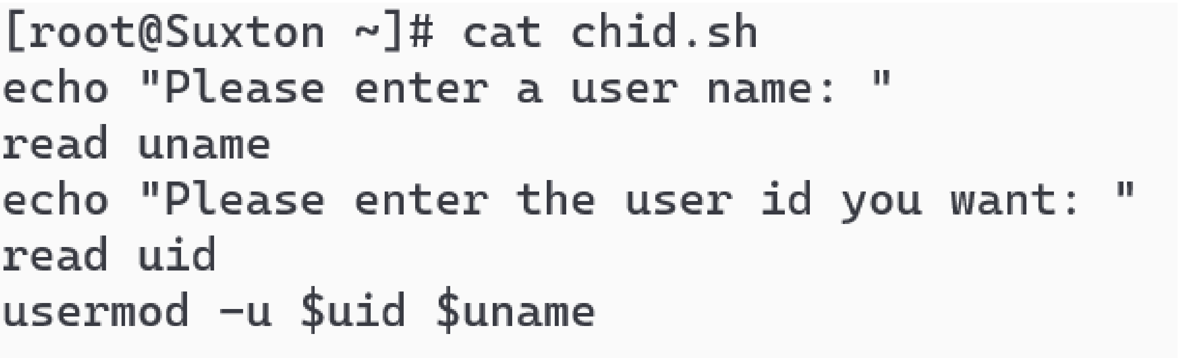
```
[root@Suxton ~]# cat /etc/passwd|grep test
test:x:1001:1001::/home/test:/bin/bash
```

3. 修改id为1002



```
[root@Suxton ~]# usermod -u 1002 test
[root@Suxton ~]# cat /etc/passwd|grep test
test:x:1002:1001::/home/test:/bin/bash
```

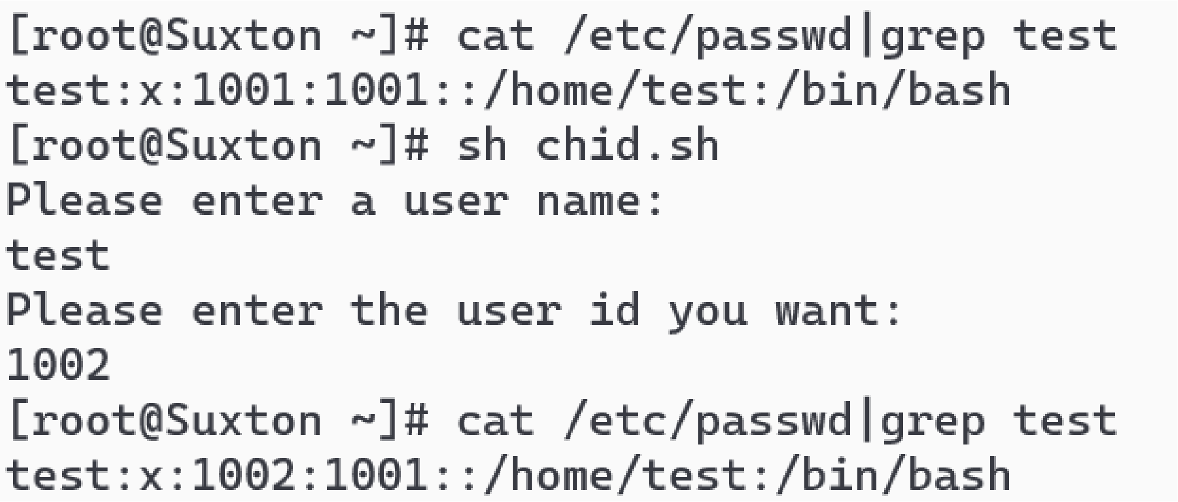
4. 将上面的功能写为shell，使用cat命令查看



```
[root@Suxton ~]# cat chid.sh
echo "Please enter a user name: "
read uname
echo "Please enter the user id you want: "
read uid
usermod -u $uid $uname
```

5. 测试.sh文件

   可以看见，id已经被成功更改



```
[root@Suxton ~]# cat /etc/passwd|grep test
test:x:1001:1001::/home/test:/bin/bash
[root@Suxton ~]# sh chid.sh
Please enter a user name:
test
Please enter the user id you want:
1002
[root@Suxton ~]# cat /etc/passwd|grep test
test:x:1002:1001::/home/test:/bin/bash
```