操作系统

实验一

姓名：

学号：

邮箱：

一、使用 Linux Shell 命令完成以下操作：

1. 查看当前登录在系统中的用户列表、系统中的用户总数和系统启动时间

当前登录在系统中的用户列表：

```
wxt@wxt:~$ w
 19:55:36 up 7 min,  1 user,  load average: 2.24, 2.20, 1.29
USER     TTY      来自             LOGIN@   IDLE   JCPU   PCPU WHAT
wxt      :0       :0               19:49    ?xdm?   1:31   0.02s /usr/lib/gdm3/gdm-x-session --run-script env GNOME_SHELL_
```

系统中的用户总数：

```
wxt@wxt:~$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
systemd-timesync:x:102:104:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:103:106::/nonexistent:/usr/sbin/nologin
syslog:x:104:110::/home/syslog:/usr/sbin/nologin
_apt:x:105:65534::/nonexistent:/usr/sbin/nologin
tss:x:106:111:TPM software stack,,,:/var/lib/tpm:/bin/false
uuidd:x:107:114::/run/uuidd:/usr/sbin/nologin
tcpdump:x:108:115::/nonexistent:/usr/sbin/nologin
avahi-autoipd:x:109:116:Avahi autoip daemon,,,:/var/lib/avahi-autoipd:/usr/sbin/nologin
usbmux:x:110:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
rtkit:x:111:117:RealtimeKit,,,:/proc:/usr/sbin/nologin
dnsmasq:x:112:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
cups-pk-helper:x:113:120:user for cups-pk-helper service,,,:/home/cups-pk-helper:/usr/sbin/nologin
speech-dispatcher:x:114:29:Speech Dispatcher,,,:/run/speech-dispatcher:/bin/false
avahi:x:115:121:Avahi mDNS daemon,,,:/var/run/avahi-daemon:/usr/sbin/nologin
kernoops:x:116:65534:Kernel Oops Tracking Daemon,,,:/:/usr/sbin/nologin
saned:x:117:123::/var/lib/saned:/usr/sbin/nologin
nm-openvpn:x:118:124:NetworkManager OpenVPN,,,:/var/lib/openvpn/chroot:/usr/sbin/nologin
hplip:x:119:7:HPLIP system user,,,:/run/hplip:/bin/false
whoopsie:x:120:125::/nonexistent:/bin/false
colord:x:121:126:colord colour management daemon,,,:/var/lib/colord:/usr/sbin/nologin
geoclue:x:122:127::/var/lib/geoclue:/usr/sbin/nologin
pulse:x:123:128:PulseAudio daemon,,,:/var/run/pulse:/usr/sbin/nologin
gnome-initial-setup:x:124:65534::/run/gnome-initial-setup/:/bin/false
gdm:x:125:130:Gnome Display Manager:/var/lib/gdm3:/bin/false
wxt:x:1000:1000:wxt,,,:/home/wxt:/bin/bash
systemd-coredump:x:999:999:systemd Core Dumper:/:/usr/sbin/nologin
```

系统启动时间：

```
wxt@wxt:~$ who -b
         系统引导 2020-10-20 19:47
wxt@wxt:~$
```

2. 将系统文件/etc/profile 复制到主用户目录，并改名为 profile.txt， 查看此文件的内容，并对非空行进行编号；重新打开此文件，从 profile 的第 5 行开始显示，每屏幕仅显示 5 行

```
wxt@wxt:~$ cp /etc/profile /home/wxt/
wxt@wxt:~$ cd /home/wxt
wxt@wxt:~$ ls
公共的  模板  视频  图片  文档  下载  音乐  桌面  profile  snap  VMwareTools-10.3.21-14772444.tar.gz  vmware-tools-distrib
wxt@wxt:~$ mv profile profile.txt
wxt@wxt:~$ ls
公共的  模板  视频  图片  文档  下载  音乐  桌面  profile.txt  snap  VMwareTools-10.3.21-14772444.tar.gz  vmware-tools-distrib
```

```
wxt@wxt:~$ cat -b profile.txt
     1  # /etc/profile: system-wide .profile file for the Bourne shell (sh(1))
     2  # and Bourne compatible shells (bash(1), ksh(1), ash(1), ...).

     3  if [ "${PS1-}" ]; then
     4    if [ "${BASH-}" ] && [ "$BASH" != "/bin/sh" ]; then
     5      # The file bash.bashrc already sets the default PS1.
     6      # PS1='\h:\w\$ '
     7      if [ -f /etc/bash.bashrc ]; then
     8        . /etc/bash.bashrc
     9      fi
    10    else
    11      if [ "`id -u`" -eq 0 ]; then
    12        PS1='# '
    13      else
    14        PS1='$ '
    15      fi
    16    fi
    17  fi

    18  if [ -d /etc/profile.d ]; then
    19    for i in /etc/profile.d/*.sh; do
    20      if [ -r $i ]; then
    21        . $i
    22      fi
    23    done
    24    unset i
    25  fi
```

```
wxt@wxt:~$ cat profile.txt | head -n 10 | tail -n +6
    # The file bash.bashrc already sets the default PS1.
    # PS1='\h:\w\$ '
    if [ -f /etc/bash.bashrc ]; then
      . /etc/bash.bashrc
    fi
```

3. 在主用户目录创建临时目录 tmp，在此目下录， 将/etc 目录压缩成 etc.zip 文件，然后解压缩

```
wxt@wxt:~$ mkdir tmp
wxt@wxt:~$ zip /home/wxt/tmp/etc.zip /etc
  adding: etc/ (stored 0%)
wxt@wxt:~$ zip -r /home/wxt/tmp/etc.zip /etc
       zip warning: name not matched: /etc/network/if-post-down.d/avahi-daemon
       zip warning: name not matched: /etc/pulse/client.conf.d/01-enable-autospawn.conf
  adding: etc/ (stored 0%)
  adding: etc/passwd (deflated 64%)
  adding: etc/rc1.d/ (stored 0%)
  adding: etc/rc1.d/K01open-vm-tools (deflated 60%)
  adding: etc/rc1.d/K01rsyslog (deflated 57%)
  adding: etc/rc1.d/K01spice-vdagent (deflated 64%)
  adding: etc/rc1.d/K01cups (deflated 58%)
  adding: etc/rc1.d/K01openvpn (deflated 69%)
  adding: etc/rc1.d/K01gdm3 (deflated 59%)
  adding: etc/rc1.d/K01whoopsie (deflated 44%)
  adding: etc/rc1.d/K01uuidd (deflated 53%)
  adding: etc/rc1.d/K01cups-browsed (deflated 56%)
  adding: etc/rc1.d/K01kerneloops (deflated 59%)
  adding: etc/rc1.d/K01avahi-daemon (deflated 63%)
  adding: etc/rc1.d/K01saned (deflated 61%)
  adding: etc/rc1.d/K01ufw (deflated 68%)
  adding: etc/rc1.d/K01speech-dispatcher (deflated 63%)
  adding: etc/rc1.d/K01irqbalance (deflated 59%)
  adding: etc/rc1.d/K01pulseaudio-enable-autospawn (deflated 42%)
```

```
wxt@wxt:~/tmp$ unzip etc.zip
```

4. 查找/etc 目录下包含字符串“ss”的文件；复制/etc/passwd 文件到用户的主目录下，搜索这个文件中包含字符串“root”的行，并显示行号

```
wxt@wxt:~/tmp$ find . -name "*ss*"
./etc/passwd
./etc/network/if-pre-up.d/wireless-tools
./etc/network/if-post-down.d/wireless-tools
./etc/ssl
./etc/ssl/certs/D-TRUST_Root_Class_3_CA_2_EV_2009.pem
./etc/ssl/certs/Sonera_Class_2_Root_CA.pem
./etc/ssl/certs/Starfield_Class_2_CA.pem
./etc/ssl/certs/SwissSign_Gold_CA_-_G2.pem
./etc/ssl/certs/VeriSign_Class_3_Public_Primary_Certification_Authority_-_G5.pem
./etc/ssl/certs/Buypass_Class_3_Root_CA.pem
./etc/ssl/certs/Verisign_Class_3_Public_Primary_Certification_Authority_-_G3.pem
./etc/ssl/certs/Certplus_Class_2_Primary_CA.pem
./etc/ssl/certs/T-TeleSec_GlobalRoot_Class_2.pem
./etc/ssl/certs/DigiCert_Assured_ID_Root_G3.pem
./etc/ssl/certs/DigiCert_High_Assurance_EV_Root_CA.pem
./etc/ssl/certs/T-TeleSec_GlobalRoot_Class_3.pem
./etc/ssl/certs/Go_Daddy_Class_2_CA.pem
./etc/ssl/certs/D-TRUST_Root_Class_3_CA_2_2009.pem
./etc/ssl/certs/ssl-cert-snakeoil.pem
./etc/ssl/certs/DigiCert_Assured_ID_Root_CA.pem
./etc/ssl/certs/SwissSign_Silver_CA_-_G2.pem
./etc/ssl/certs/NetLock_Arany_=Class_Gold=_Főtanúsítvány.pem
./etc/ssl/certs/DigiCert_Assured_ID_Root_G2.pem
./etc/ssl/certs/VeriSign_Class_3_Public_Primary_Certification_Authority_-_G4.pem
./etc/ssl/certs/Buypass_Class_2_Root_CA.pem
./etc/ssl/openssl.cnf
```

```
wxt@wxt:~/tmp$ cp /etc/passwd /home/wxt
wxt@wxt:~/tmp$ cd ..
wxt@wxt:~$ ls
Desktop  Documents  Downloads  Music  passwd  Pictures  Public  Templates  tmp  Videos
wxt@wxt:~$ grep -w -c "root" passwd
1
```

5. 创建一个新用户 user1，给该用户设置密码为 LoveLinux，将用户名更改为 user2。 创建 user3,将 user3 的有效组切换为 admin。 切换到 user3，在/home 目录下创建 dir 目录。 切换到 user2， 查看 user2 是否可以在 dir 目录下创建、删除文件。如果不可以修改这个目录的权限，或者修改这个目录的所有者、所属组，使得用户 user2 可以在这个目录下创建、删除文件。

user1：

```
wxt@wxt:~$ sudo adduser user1
[sudo] password for wxt:
Adding user `user1' ...
Adding new group `user1' (1001) ...
Adding new user `user1' (1001) with group `user1' ...
Creating home directory `/home/user1' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for user1
Enter the new value, or press ENTER for the default
        Full Name []:
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] Y
```

```
wxt@wxt:~$ sudo usermod -l user2 user1
wxt@wxt:~$ cat /etc/passwd|grep user2
user2:x:1001:1001:,,,:/home/user1:/bin/bash
wxt@wxt:~$
```

user3:

```
wxt@wxt:~$ sudo groupadd admin
```

```
wxt@wxt:~$ sudo gpasswd -a user3 admin
Adding user user3 to group admin
```

```
wxt@wxt:~$ grep "user3" /etc/group
user3:x:1002:
admin:x:1003:user3
```

```
user3@wxt:/home/wxt$ sudo newgrp admin
[sudo] password for user3:
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
```

```
root@wxt:/home# mkdir dir
```

```
root@wxt:/home# su user2
user2@wxt:/home$ cd dir
user2@wxt:/home/dir$ mkdir createdir
mkdir: cannot create directory ‘createdir’: Permission denied
```

user2 不能在 dir 目录下创建、删除文件，修改 dir 所有者后，user2 可以在这个目录下创建、删除文件

```
user2@wxt:/home/dir$ su user3
Password:
user3@wxt:/home/dir$ sudo chown -R user2 /home/dir
user3@wxt:/home/dir$ su user2
Password:
user2@wxt:/home/dir$ mkdir createdir
user2@wxt:/home/dir$ ls
createdir
user2@wxt:/home/dir$ rm -r createdir
user2@wxt:/home/dir$ ls
```

6. 完全使用命令下载、 安装、运行并卸载 Linux 版本的 QQ

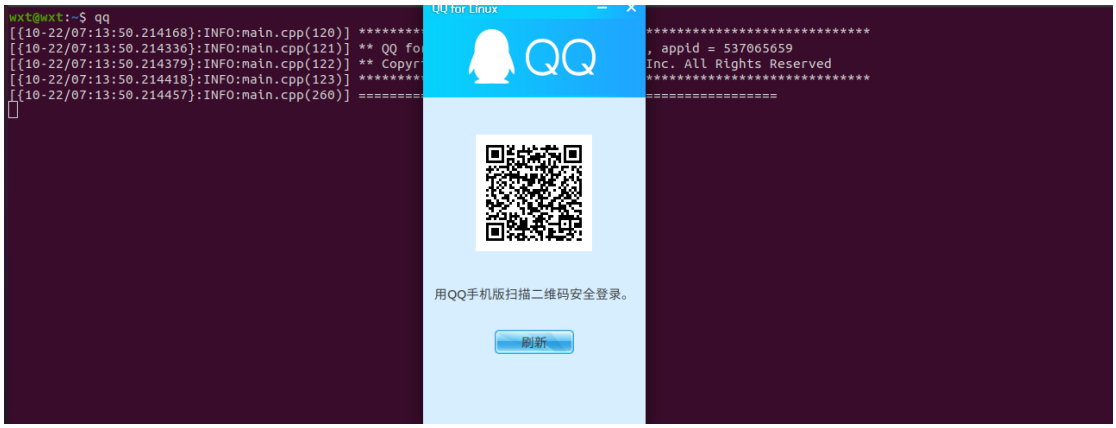
下载：

```
wxt@wxt:~$ wget http://down.qq.com/qqweb/LinuxQQ/linuxqq_2.0.0-b2-1084_amd64.deb
--2020-10-22 06:56:29--  http://down.qq.com/qqweb/LinuxQQ/linuxqq_2.0.0-b2-1084_amd64.deb
Resolving down.qq.com (down.qq.com)... 124.14.21.21, 124.14.21.14, 124.14.21.22, ...
Connecting to down.qq.com (down.qq.com)|124.14.21.21|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 12400036 (12M) [application/octet-stream]
Saving to: ‘linuxqq_2.0.0-b2-1084_amd64.deb’

linuxqq_2.0.0-b2-1084_amd64.deb     100%[=====================================================================>]  11.83M  7.47MB/s    in 1.6s

2020-10-22 06:56:31 (7.47 MB/s) - ‘linuxqq_2.0.0-b2-1084_amd64.deb’ saved [12400036/12400036]
```

安装：

```
wxt@wxt:~$ sudo dpkg -i linuxqq_2.0.0-b2-1084_amd64.deb
Selecting previously unselected package linuxqq.
(Reading database ... 179004 files and directories currently installed.)
Preparing to unpack linuxqq_2.0.0-b2-1084_amd64.deb ...
Unpacking linuxqq (2.0.0-b2) ...
Setting up linuxqq (2.0.0-b2) ...
Processing triggers for gnome-menus (3.36.0-1ubuntu1) ...
Processing triggers for desktop-file-utils (0.24-1ubuntu2) ...
Processing triggers for mime-support (3.64ubuntu1) ...
```

运行：



卸载：

```
Killed
wxt@wxt:~$ sudo dpkg -r linuxqq
[sudo] password for wxt:
(Reading database ... 179017 files and directories currently installed.)
Removing linuxqq (2.0.0-b2) ...
dpkg: warning: while removing linuxqq, directory '/usr/local/share' not empty so not removed
dpkg: warning: while removing linuxqq, directory '/usr/local/lib' not empty so not removed
Processing triggers for gnome-menus (3.36.0-1ubuntu1) ...
Processing triggers for desktop-file-utils (0.24-1ubuntu2) ...
Processing triggers for mime-support (3.64ubuntu1) ...
wxt@wxt:~$
```

7. 查看网络适配器的网络设置，将 dhcp 动态 IP 的设置方式改为 static 静态 IP 的设置方式；查看当前系统服务端口的监听状态。

```
wxt@wxt:~$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.192.129  netmask 255.255.255.0  broadcast 192.168.192.255
        inet6 fe80::77da:8429:2ff1:a082  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:fd:56:d3  txqueuelen 1000  (Ethernet)
        RX packets 62345  bytes 90202098 (90.2 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 6181  bytes 508322 (508.3 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 500  bytes 47347 (47.3 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 500  bytes 47347 (47.3 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

```
wxt@wxt:/etc/netplan$ sudo vim /etc/netplan/01-network-manager-all.yaml
```

```
# Let NetworkManager manage all devices on this system
network:
  version: 2
  renderer: NetworkManager
  ethernets:
          ens33:
                  dhcp4: false
                  addresses: [192.168.43.142/24]
                  gateway4: 192.168.43.1
                  nameservers:
                          addresses: [192.168.43.1,8.8.8.8]
~
```

```
wxt@wxt:/etc/netplan$ sudo netplan --debug apply
** (generate:8107): DEBUG: 08:08:59.698: Processing input file /etc/netplan/01-network-manager-all.yaml..
** (generate:8107): DEBUG: 08:08:59.698: starting new processing pass
** (generate:8107): DEBUG: 08:08:59.698: We have some netdefs, pass them through a final round of validation
** (generate:8107): DEBUG: 08:08:59.698: ens33: setting default backend to 2
** (generate:8107): DEBUG: 08:08:59.699: Configuration is valid
** (generate:8107): DEBUG: 08:08:59.699: Generating output files..
** (generate:8107): DEBUG: 08:08:59.699: networkd: definition ens33 is not for us (backend 2)
(generate:8107): GLib-DEBUG: 08:08:59.699: posix_spawn avoided (fd close requested)
DEBUG:no netplan generated networkd configuration exists
DEBUG:netplan generated NM configuration changed, restarting NM
DEBUG:ens33 not found in {}
DEBUG:Merged config:
network:
  bonds: {}
  bridges: {}
  ethernets:
    ens33:
      addresses:
      - 192.168.43.142/24
      dhcp4: false
      gateway4: 192.168.43.1
      nameservers:
        addresses:
        - 192.168.43.1
        - 8.8.8.8
  vlans: {}
  wifis: {}
```

静态 ip 设置好了：

```
wxt@wxt:/etc/netplan$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.43.142  netmask 255.255.255.0  broadcast 192.168.43.255
        inet6 fe80::20c:29ff:fefd:56d3  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:fd:56:d3  txqueuelen 1000  (Ethernet)
        RX packets 71764  bytes 102710284 (102.7 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 9033  bytes 705696 (705.6 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

```
PING 192.168.43.142 (192.168.43.142) 56(84) bytes of data.
64 bytes from 192.168.43.142: icmp_seq=1 ttl=64 time=0.020 ms
64 bytes from 192.168.43.142: icmp_seq=2 ttl=64 time=0.041 ms
64 bytes from 192.168.43.142: icmp_seq=3 ttl=64 time=0.030 ms
64 bytes from 192.168.43.142: icmp_seq=4 ttl=64 time=0.057 ms
64 bytes from 192.168.43.142: icmp_seq=5 ttl=64 time=0.040 ms
64 bytes from 192.168.43.142: icmp_seq=6 ttl=64 time=0.036 ms
```

8. 插入 u 盘，在/mnt 下建立一个名叫 USB 的文件夹,然后将 u 盘挂载到 /mnt/USB 下，在此目录下创建一个 temp.txt 文件，然后卸载 u 盘

```
wxt@wxt:/mnt$ sudo mkdir USB
wxt@wxt:/mnt$ ls
USB
```

```
wxt@wxt:/mnt$ sudo fdisk -l
```

```
Disk /dev/sdb: 14.5 GiB, 15552479232 bytes, 30375936 sectors
Disk model: Ultra
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x3ac4a612

Device     Boot   Start      End  Sectors Size Id Type
/dev/sdb1  *    1142784 30375935 29233152  14G  c W95 FAT32 (LBA)
```

/dev/sdb1 即要挂载的 u 盘，所以：

```
wxt@wxt:/mnt$ mount /dev/sdb1 /mnt/USB
mount: only root can do that
wxt@wxt:/mnt$ sudo mount /dev/sdb1 /mnt/USB
wxt@wxt:/mnt$ cd USB
wxt@wxt:/mnt/USB$ ls
'RationalRose Enterprise Edition.pdf'  'System Volume Information'   毛概报告.docx
 RationalRose.rar                       virtualdrivemaster.exe       练习2.pdf
```

```
wxt@wxt:/mnt/USB$ vim temp.txt
wxt@wxt:/mnt/USB$ ls
'RationalRose Enterprise Edition.pdf'   temp.txt
 RationalRose.rar                       virtualdrivemaster.exe
'System Volume Information'             毛概报告.docx
```

```
wxt@wxt:/mnt/USB$ sudo umount /mnt/USB
```

三、 Makefile 实验

1. 请自行查找资料，阅读 Makefile 相关资料， 了解 Makefile 的基本概念和基本结构，初步掌握编写简单 Makefile 的方法，了解递归 Make 的编译过程，初步掌握利用 GNU Make 编译应用程序的方法。推荐阅读：makefile 教程（中文版）， 陈皓著

2. （1）利用文本编辑器（vi） 编写一种排序算法 sort.c，对一个数组中的整数进行排序;

```c
#include <stdio.h>

int a[1000] = {};

void bubble_sort(int n){
        int t=0;

        for(int i=0;i<n-1; i++){
                for(int j=0; j<n-i-1; j++){
                        if(a[j]>a[j+1]){
                                t = a[j];
                                a[j] = a[j+1];
                                a[j+1] = t;
                        }
                }
        }

        for(int i=0; i<n; i++)
                printf("%d\t", a[i]);
}

int main(){
        int n=0;
        scanf("%d", &n);
        for(int i=0; i<n; i++)
                scanf("%d", a+i);
        bubble_sort(n);
        return 0;
```

（2）利用 gcc 手动编译、运行该程序；

```
wxt@wxt:~/tmp$ gcc -Wall -ggdb -o mysort sort.c
```

```
wxt@wxt:~/tmp$ ./mysort
10
3 6 7 9 0 2 8 5 1 4
0       1       2       3       4       5       6       7       8       9
```

（3）利用 gdb 手动加入断点进行调试，在屏幕上打印断点信息，以及任何一个变量的值。

```
(gdb) b bubble_sort
Breakpoint 1 at 0x1189: file sort.c, line 5.
(gdb) r
Starting program: /home/wxt/tmp/mysort
5
4 5 2 1 3

Breakpoint 1, bubble_sort (n=21845) at sort.c:5
5       void bubble_sort(int n){
(gdb) l
1       #include <stdio.h>
2
3       int a[1000] = {};
4
5       void bubble_sort(int n){
6               int t=0;
7
8               for(int i=0; i<n-1; i++){
9                       for(int j=0; j<n-i-1; j++){
10                              if(a[j]>a[j+1]){
```

```
(gdb) p n
$4 = 5
(gdb) p a
$5 = {4, 5, 2, 1, 3, 0 <repeats 995 times>}
```

3. 针对 sort.c 利用文本编辑器创建一个 makefile 文件，通过 make 编译次程序，并运行。

```
sort: sort.o
        gcc sort.o -o sort

sort.o: sort.c
        gcc -c sort.c -o sort.o

clean:
        rm sort sort.o
```

```
wxt@wxt:~/tmp$ make
gcc -c sort.c -o sort.o
gcc sort.o -o sort
wxt@wxt:~/tmp$ ./sort
5
2
3
4
5
1
1       2       3       4       5       wxt@wxt:~/tmp$
```

4. (1) 修改 sort.c，在排序完成后创建一个进程；

```
void myfork(){
        pid_t pid;
        pid = fork();

        if(pid == -1){
                perror("fork failed\n");
                exit(1);
        }
        else if(pid == 0){
                printf("child process pid=%d, ppid=%d\n", getpid(), getppid());
        }
        else {
                printf("father process pid=%d, ppid=%d, child %d\n", getpid(), getppid(), pid);
                sleep(1);
        }
}
```

```
wxt@wxt:~/tmp$ vim sort.c
wxt@wxt:~/tmp$ make
gcc -c sort.c -o sort.o
gcc sort.o -o sort
wxt@wxt:~/tmp$ ./sort
5
2 4 3 5
1
1       2       3       4       5
father process pid=9657, ppid=9527, child 9658
child process pid=9658, ppid=9657
```

(2) 创建完成后父进程打印有序队列的首地址，然后休眠 5 秒钟；

```
void myfork(){
        pid_t pid;
        pid = fork();

        if(pid == -1){
                perror("fork failed\n");
                exit(1);
        }
        else if(pid == 0){
                printf("child process pid=%d, ppid=%d\n", getpid(), getppid());
        }
        else {
                printf("father process pid=%d, ppid=%d, child %d\n", getpid(), getppid(), pid);
                printf("%p\n",a);
                //sleep(1);
                sleep(5);
        }
}
```

```
wxt@wxt:~/tmp$ ./sort
5 3 4 1 2 5
1       2       3       4       5
father process pid=9894, ppid=9527, child 9895
0x5638a9409040
child process pid=9895, ppid=9894
1 2 3 4 5 9
1 2 3 4 5 9
wxt@wxt:~/tmp$
```

(3) 子进程调用一个在 insert.c 中实现的插入函数，在有序队列中插入一个整数，然后打印队列的首地址。

简单一点，直接在 sort.c 的数组最后插入一个 9：

```
extern int a[];

int insert(int x, int n){
        a[n]=x;
        n++;
        return n;
}
~
```

(4) 针对 sort.c 和 insert.c 利用文本编辑器创建一个 makefile 文件，通过 make 编译此程序，并运行。

```
sort: sort.o insert.o
        gcc sort.o insert.o -o sort

sort.o: sort.c
        gcc -c sort.c -o sort.o

insert.o: insert.c
        gcc -c insert.c -o insert.o

clean:
        rm sort sort.o insert.o
~
```

```
wxt@wxt:~/tmp$ ./sort
5
3 2 4 5 1
1       2       3       4       5
father process pid=9950, ppid=9527, child 9951
0x55d00be59040
child process pid=9951, ppid=9950
0x55d00be59040
1 2 3 4 5 9
```

(5) 分析运行结果，写出你的发现。

执行 fork()语句后，操作系统创建子进程，执行完成后，子进程返回 0，父进程返回子进程 pid。

5. 阅读 Linux 源码中的/Documentation/kbuild/makefiles.txt 文件（网上有中文版），并根据此文档分析并注释/kernel 目录下的 Makefile 文件。

Linux 内核中的 Makefile 以及与 Makefile 直接相关的文件有：
Makefile：顶层 Makefile，是整个内核配置、编译的总体控制文件。
.config：内核配置文件，包含由用户选择的配置选项，用来存放内核配置后的结果（如 make config）。
arch/*/Makefile：位于各种 CPU 体系目录下的 Makefile，如 arch/arm/Makefile，是针对特定平台的 Makefile。
各个子目录下的 Makefile：比如 drivers/Makefile，负责所在子目录下源代码的管理。
Rules.make：规则文件，被所有的 Makefile 使用。

obj-y 用来定义哪些文件被编进内核：
obj-y 中定义的.o 文件由当前目录下的.c 或.S 文件编译生成，它们连同下级子目录的 built-in.o 文件一起被组合成(使用“$(LD)-R”命令)当前目录下的 built-in.o 文件，这个 built-in.o
文件被它的上一层 Makefile 使用

obj-m 用来定义哪些文件被编译成可加载模块：
obj-m 中定义的.o 文件由当前目录下的.c 或.S 文件编译生成,它们被编译成模块，一个模块可以由一个或几个.o 文件组成，对于有多个源文件的模块，除在

obj-m 中增加一个.o 文件外，还要定义一个<module_name>-objs 变量来告诉 Makefile 这个.o 文件由哪些

lib-y 用来定义哪些文件被编成库文件：
liy-y 中定义的.o 文件由当前目录下的.c 或.S 文件编译生成，它们被打包成当前目录下的一个库文件：lib.a ；同时出现在 obj-y 和 lib-y 中的.o 文件，不会被包含进 lib.a 中，要把这个 lib.a 编译进内核中，需要在顶层 Makefile 中 libs-y 变量中列出当前目录

obj-y、obj-m 还可以指定要进入的下一层子目录：
obj-$(CONFIG_JFFS2_FS) += jffs2/

怎样编译这些文件：
1.全局的
适用于整个内核代码，在顶层 Makefile 和 arch/$(ARCH)/Makefile 中定义，这些选项的名称为 CFLAGS(编译 C 文件的选项)、AFLAGS(编译汇编文件的选项)、LDFLAGS(连接文件的选项)、 ARFLAGS(制作库文件的选项)
2.局部的
在各个子目录中定义，针对当前 Makefile 中的所有文件，名称分别为：
EXTRA_CFLAGS、EXTRA_AFLAGS、EXTRA_LDFLAGS、EXTRA_ARFLAGS
3.个体的
仅适用于某个文件，如果想针对某个文件定义它的编译选项，可以使用 CFLAGS_$@,AFLAGS_$@，前者用于编译某个 C 文件，后者用于编译某个汇编文件。$@表示某个目录文件

怎样连接这些文件：
在顶层 Makefile 中，目录名的后面直接加上 built-in.o 或 lib.a，表示要连接进内核的文件，如下所示：

```
init-y := $(patsubst %/, %/built-in.o, $(init-y))
core-y := $(patsubst %/, %/built-in.o, $(core-y))
drivers-y :=$(patsubst %/, %/built-in.o, $(drivers-y))
net-y := $(patsubst %/, %/built-in.o, $(net-y))
libs-y1 := $(patsubst %/, %/lib.a, $(libs-y))
libs-y2 := $(patsubst %/, %/built-in.o, $(libs-y))
libs-y := $(libs-y1) $(libs-y2)
```