

上海科技大学 GPU 集群

登陆说明

协管员

2018 年 4 月

目录

1 使用集群进行计算业务	1
1.1 系统登陆	1
1.1.1 登陆 IP 地址	1
1.1.2 命令行终端登录	1
1.1.3 文件上传下载	2
1.2 编译、安装 OpenMPI 示例	3
1.3 环境变量配置	4
1.4 命令行运行程序	4
1.4.1 运行串行程序	4
1.5 使用作业调度（命令行模式）运行程序	5
1.5.1 PBS 的基本命令	5
1.5.2 查询队列信息	6
1.5.3 查询节点信息 <code>pestat</code>	6
1.5.4 查询作业运行状态	7
1.5.5 删除作业	8
1.5.6 作业调度系统使用举例	9
2 附录—Linux 常用命令	13
1.1 浏览目录命令	13
1.2 浏览文件命令	13
1.3 目录操作命令	14
1.4 文件操作命令	15
1.5 查找类命令	16
1.6 用法帮助命令	17
1.7 打包、解包，压缩、解压缩命令	17
1.8 时间相关命令	19
1.9 系统信息类命令	19
1.10 网络通讯类命令	20
1.11 软件包管理命令	21
1.12 编辑器命令（ <code>vim</code> ）	22
1.13 用户管理命令	24
1.14 用户组管理命令	24
1.15 输入/输出重定向与管道命令	24

1 使用集群进行计算业务

1.1 系统登陆

1.1.1 登陆 IP 地址

登陆节点 login，用户可以通过多种登录方式登录集群系统。

1.1.2 命令行终端登录

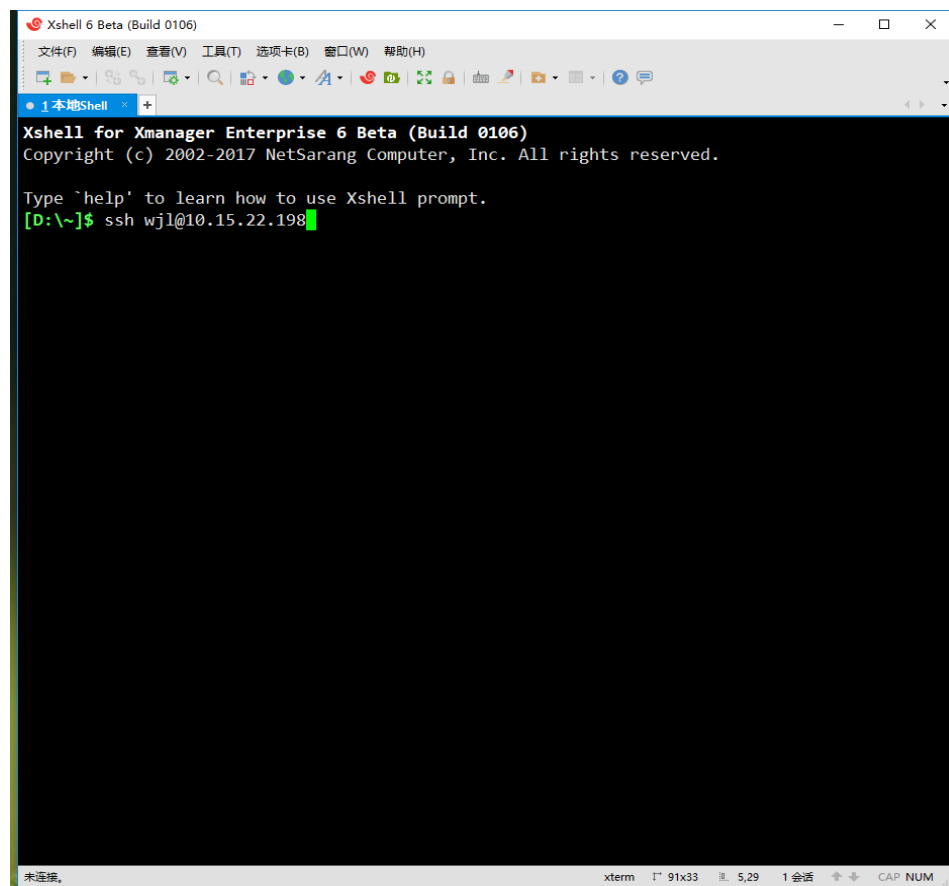
Windows 用户可以用 SSH Secure Shell Client，PuTTY，SecureCRT，Xshell 等 SSH 客户端软件登录。推荐使用 Xshell。

下载安装 xshell 客户端软件，登陆方法如下：

1. 双击 xshell 图标：



2. 启动控制台，输入“用户名@10.15.22.198”



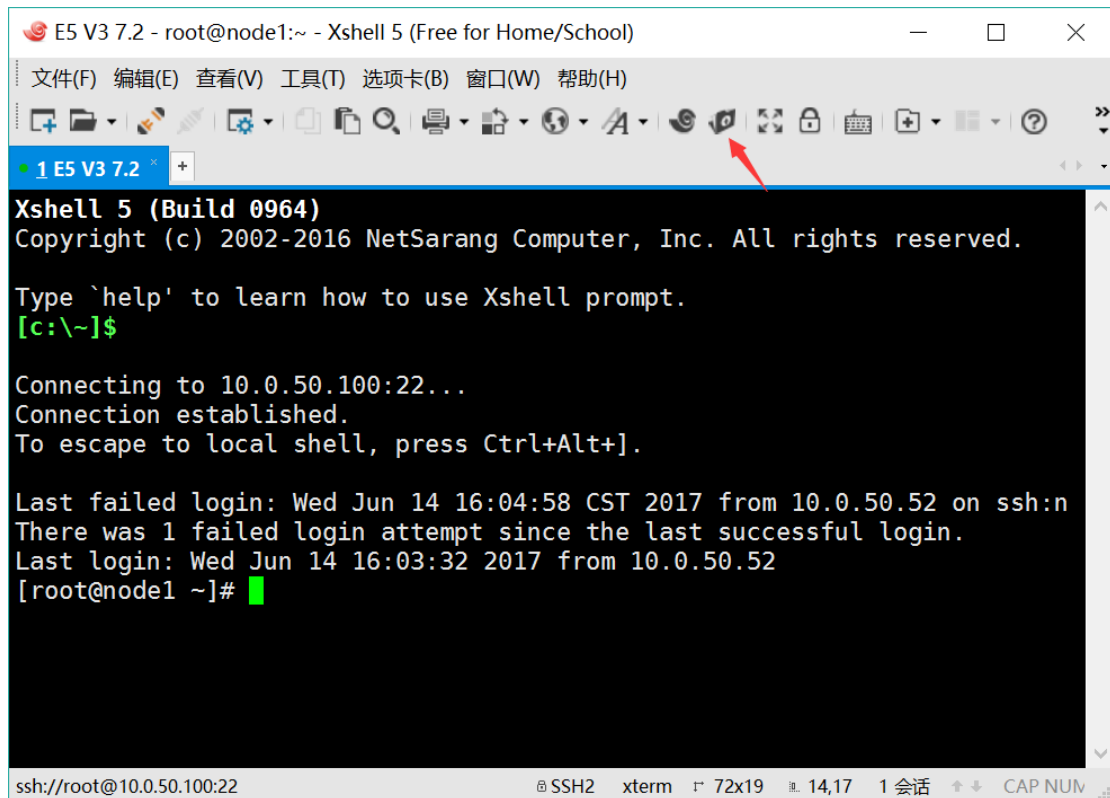
3. 用户名和密码验证无误后，即可登陆到集群。

Linux 用户可以直接在命令行终端中执行 `ssh` 命令进行登录:

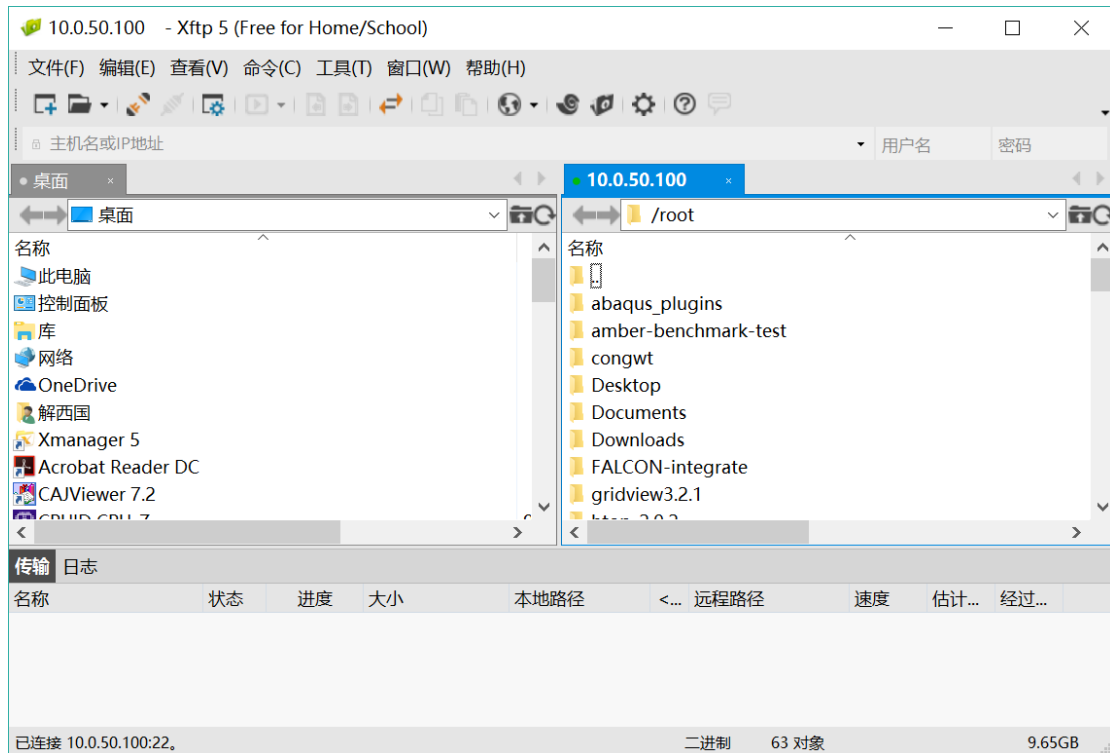
```
$ ssh yourname@10.15.22.198 (登录节点 IP 地址)
```

1.1.3 文件上传下载

Windows 用户可以用 SSH Secure Shell Client, winscp, Xftp 等软件实现文件的上传下载。推荐使用 Xftp。安装好 Xftp 后, 可以直接在 Xshell 中开启文件传输, Xshell 会自动启动 Xftp。



Xshell 自动启动 Xftp



Linux 操作系统直接使用命令

```
scp filename test@ip:/home/test
```

1.2 编译、安装 OpenMPI 示例

以 OpenMPI 1.6.3 为例：

```
$ tar zxvf openmpi-2.0.1.tar.gz
$ cd openmpi-2.0.1
$ ./configure --prefix=/public/software/mpi/openmpi-2.0.1-intel
--enable-mpirun-prefix-by-default --without-psm CC=icc CXX=icpc FC=ifort
F77=ifort
$ make -j 8 && make install
```

设置环境变量脚本：

```
vim /public/sist/software/profile.d/openmpi-intel-env.sh

#!/bin/bash

export MPI_HOME=/public/sist/software/mpi/openmpi-2.0.1-intel

export PATH=${MPI_HOME}/bin:${PATH}
```

```
export LD_LIBRARY_PATH=${MPI_HOME}/lib:${LD_LIBRARY_PATH}

export MANPATH=${MPI_HOME}/share/man:${MANPATH}
```

1.3 环境变量配置

同一工作组或基础环境软件可以安装在共享目录下，避免每人单独安装浪费时间。

规范软件安装路径是：/public/software/

规范软件配置文件路径是：/public/sist/software/profile.d/

以 CUDA 为例，本集群部署了 cuda8.0 和 cuca9.1。

“which nvcc”命令可查询 cuda 所在路径，如有异常提示请手动加载 cuda 环境变量，方法一为每次登陆需要加载；方法二为登陆系统自动加载。

方法一：“source /public/sist/profile.d/cudaXXX.sh”（XXX 为 cuda 的版本）

cuda8.0 和 cuda9.1 安装路径是：/public/software/cudaXXX

cuda8.0 和 cuda9.1 配置文件路径是：/public/sist/software/profile.d/cudaXXX.sh

“nvcc -V”命令可验证所加载的 cuda 版本。

“cat /public/sist/software/profile.d/xxxxsoft.sh”可直接列出软件实际安装目录。

方法二：

```
echo "export PATH=/public/software/cuda-9.1/bin:$PATH" >> ~/.bashrc
```

```
echo "export LD_LIBRARY_PATH=/public/software/cuda-9.1/lib64:$LD_LIBRARY_PATH" >> ~/.bashrc
```

```
source ~/.bashrc
```

1.4 命令行运行程序

1.4.1 运行串行程序

方法一

```
cd /home/your_account/your_workdir

./your_code
```

方法二

```
cd $HOME

vi .bashrc
```

添加

```
export PATH=/home/your_account/your_workdir:$PATH
```

执行命令

```
your_code
```

1.5 使用作业调度（命令行模式）运行程序

1.5.1 PBS 的基本命令

在 PBS 系统中，用户使用 qsub 命令提交用户程序。

```
qsub xxx.pbs
```

用户运行程序的命令及 PBS 环境变量设置组成 PBS 作业脚本，提交格式如下：

注释,以“#”开头

PBS 指令,以“#PBS”开头

示例 OpenMPI 脚本: openmpi.pbs

```
#PBS -N openmpi

#PBS -l nodes=1:ppn=8

#PBS -j oe

#PBS -l walltime=2:00:00

cd $PBS_O_WORKDIR

echo my job id is $PBS_JOBID | tee openmpi.log
echo run nodes is following: | tee -a openmpi.log
cat $PBS_NODEFILE | tee -a openmpi.log

echo begin time is `date` | tee -a openmpi.log
id=`echo $PBS_JOBID|awk -F. '{print $1}'`
NP=`cat $PBS_NODEFILE|wc -l`

mpirun -np $NP -hostfile $PBS_NODEFILE --mca orte_rsh_agent ssh --mca btl
self,openib,sm ./program 2>&1 | tee -a openmpi.log

echo end time is `date` | tee -a openmpi.log
```

注意：算例规模的大小合理估算所需的 walltime 和 Mem，把其写进作业脚本里，这样有助于更快、

更有效地分配资源；

1.5.2 查询队列信息

```
qmgr -c 'p s'
```

以 gpu 节点为例：

```
#
# Create and define queue gpu
#
create queue gpu
set queue gpu queue_type = Execution
set queue gpu resources_max.nodect = 8
set queue gpu resources_max.walltime = 120:00:00
set queue gpu resources_default.neednodes = gpu
set queue gpu max_user_run = 2
set queue gpu enabled = True
set queue gpu started = True
```

```
qmgr -c "set queue gpu acl_users += guest"
```

添加可使用该队列的用户 guest

1.5.3 查询节点信息 pestat

```
pestat |more
```

如下输出

node	state	load	pmem	ncpu	mem	resi	usrs	tasks	jobids/users
a110	busy*	12.00	12001	12	12001	4605	4/3	12	8904 hcding
a111	busy*	12.00	12001	12	12001	4842	3/3	12	8904 hcding
a112	free	0.00	12001	12	12001	1182	1/1	0	
a113	busy*	12.00	12001	12	12001	4753	2/2	12	8904 hcding
a114	free	0.00	12001	12	12001	962	2/2	0	
a115	busy*	12.00	12001	12	12001	1473	4/3	12	9537 john
a116	free	0.00	12001	12	12001	1156	3/3	0	

a117	busy*	12.00	12001	12	12001	5090	3/3	12	8904	hcding
a118	free	0.00	12001	12	12001	2606	2/2	0		
a119	busy*	12.12	12001	12	12001	1674	4/3	12	9410	wxxu
a120	busy*	12.00	12001	12	12001	1616	4/4	12	9410	wxxu
a121	excl	11.91	12001	12	12001	2034	5/4	12	9503	john
a122	free	0.07	12001	24	12001	2181	1/1	0		
a123	busy*	12.00	12001	12	12001	1625	3/3	12	9410	wxxu
a124	free	0.01	12001	12	12001	2389	2/2	0		
a125	busy*	12.00	12001	12	12001	1652	4/4	12	9410	wxxu
a126	free	0.00	12001	12	12001	2442	1/1	0		
a127	excl	11.39*	12001	12	12001	1898	3/3	12	9503	john
a128	free	0.00	12001	12	12001	1032	1/1	0		
a129	excl	11.68	12001	12	12001	1712	3/3	12	9503	john
a130	free	1.75*	12001	12	12001	3264	5/3	3	8216	wxxu 9187 wxxu 9533
wxxu										

节点状态

- excl：所有 CPU 资源已被占用；
- busy：CPU 已接近满负荷运行；
- free：全部或部分 CPU 空闲；
- offl：管理员手动指定离线状态；

1.5.4 查询作业运行状态

qstat -an more										
如下输出：										
mgmt:										
Req'd Req'd										
Elap										
Job ID	Username	Queue	Jobname	SessID	NDS	TSK	Memory	Time		

```
S Time
-----
-----
7010.mgmt          ymei      itcs      ga88_remd1_zj      22516      8  96      --
720:0 R 231:2

a330/11+a330/10+a330/9+a330/8+a330/7+a330/6+a330/5+a330/4+a330/3+a330/2
+a330/1+a330/0+a331/11+a331/10+a331/9+a331/8+a331/7+a331/6+a331/5+a331/4
+a331/3+a331/2+a331/1+a331/0+a332/11+a332/10+a332/9+a332/8+a332/7+a332/6
+a332/5+a332/4+a332/3+a332/2+a332/1+a332/0+a334/11+a334/10+a334/9+a334/8
+a334/7+a334/6+a334/5+a334/4+a334/3+a334/2+a334/1+a334/0+a335/11+a335/10
+a335/9+a335/8+a335/7+a335/6+a335/5+a335/4+a335/3+a335/2+a335/1+a335/0
+a337/11+a337/10+a337/9+a337/8+a337/7+a337/6+a337/5+a337/4+a337/3+a337/2
+a337/1+a337/0+a339/11+a339/10+a339/9+a339/8+a339/7+a339/6+a339/5+a339/4
+a339/3+a339/2+a339/1+a339/0+a435/11+a435/10+a435/9+a435/8+a435/7+a435/6
+a435/5+a435/4+a435/3+a435/2+a435/1+a435/0
```

查询作业命令 `qstat [参数]`，其中参数可为：

- q : 列出系统队列信息
- B : 列出 PBS 服务器的相关信息
- Q : 列出队列的一些限制信息
- an: 列出队列中的所有作业及其分配的节点
- r : 列出正在运行的作业
- f jobid : 列出指定作业的信息
- Qf queue: 列出指定队列的所有信息

1.5.5 删除作业

作业删除命令: `qdel 作业号`

注意事项

- 1、非 root 用户只能查看、删除自己提交的作业；

2、**强制删除作业**，当某些作业由于节点死机无法删除时，可由 **root** 用户登录，使用 **qdel -p 作业号** 来强制删除作业

1.5.6 作业调度系统使用举例

3.6.6.1 串行作业

```
#!/bin/bash -x

#PBS -N serial

#PBS -l nodes=1:ppn=1

#PBS -l walltime=60:00:00

#PBS -j oe

#PBS -q serial

#

#define variables

#

echo "This jobs is "$PBS_JOBID@"$PBS_QUEUE

cd ${PBS_O_WORKDIR}

date

sleep 100

hostname

date
```

#PBS -l nodes=1:ppn=1 表示申请 1 个节点上的 1 颗 CPU。

#PBS -q serial 表示提交到集群上的 **serial** 队列。

3.6.6.2 并行作业

- openmpi

并行作业的脚本以 cpi 为例。示例 OpenMPI 脚本: openmpi.pbs

```
#PBS -N openmpi

#PBS -l nodes=1:ppn=8

#PBS -j oe

#PBS -l walltime=2:00:00

cd $PBS_O_WORKDIR

echo my job id is $PBS_JOBID | tee openmpi.log

echo run nodes is following: | tee -a openmpi.log

cat $PBS_NODEFILE | tee -a openmpi.log

echo begin time is `date` | tee -a openmpi.log

id=`echo $PBS_JOBID|awk -F. '{print $1}'`

NP=`cat $PBS_NODEFILE|wc -l`

mpirun -np $NP -hostfile $PBS_NODEFILE --mca orte_rsh_agent ssh --mca btl
self,openib,sm ./program 2>&1 | tee -a openmpi.log

echo end time is `date` | tee -a openmpi.log
```

- mpich2

示例 MPICH2 脚本: mpich2.pbs

```
#PBS -N mpich2

#PBS -l nodes=1:ppn=8

#PBS -j oe

#PBS -l walltime=2:00:00
```

```

cd $PBS_O_WORKDIR

echo my job id is $PBS_JOBID | tee mpich2.log
echo run nodes is following: | tee -a mpich2.log
cat $PBS_NODEFILE | tee -a mpich2.log

echo begin time is `date` | tee -a mpich2.log
id=`echo $PBS_JOBID|awk -F. '{print $1}'`
NP=`cat $PBS_NODEFILE|wc -l`

mpiexec.hydra -n $NP -launcher ssh -f $PBS_NODEFILE ./program 2>&1 | tee -a
mpich2.log

echo end time is `date` | tee -a mpich2.log

```

- mvapich2

MVAPICH2 示例脚本: mvapich2.pbs

```

#PBS -N mvapich2

#PBS -l nodes=1:ppn=8

#PBS -j oe

#PBS -l walltime=2:00:00

cd $PBS_O_WORKDIR

echo my job id is $PBS_JOBID | tee mvapich2.log
echo run nodes is following: | tee -a mvapich2.log
cat $PBS_NODEFILE | tee -a mvapich2.log

echo begin time is `date` | tee -a mvapich2.log
id=`echo $PBS_JOBID|awk -F. '{print $1}'`
NP=`cat $PBS_NODEFILE|wc -l`

```

```

mpiexec.hydra -n $NP -launcher ssh -f $PBS_NODEFILE cpi 2>&1 | tee -a mvapich2.log

echo end time is `date` | tee -a mvapich2.log

```

- `intelmpi.pbs`

IntelMPI 示例脚本: `intelmpi.pbs`

```

#PBS -N intelmpi

#PBS -l nodes=1:ppn=8

#PBS -j oe

#PBS -l walltime=2:00:00

cd $PBS_O_WORKDIR

echo my job id is $PBS_JOBID | tee intelmpi.log
echo run nodes is following: | tee -a intelmpi.log
cat $PBS_NODEFILE | tee -a intelmpi.log

echo begin time is `date` | tee -a intelmpi.log

id=`echo $PBS_JOBID|awk -F. '{print $1}'`
NP=`cat $PBS_NODEFILE|wc -l`

qstat -f1 $PBS_JOBID | grep exec_host | awk '{print $3}' | tr '+/' '\n ' | awk -v
prog="cpi " -v net="" 'NR==1 {node=$1; count=1; core[count]=$2; next}; $1==node
{count++; core[count]=$2}; $1!=node {printf("-n %d -host %s -env %s -env
I_MPI_PIN_PROCESSOR_LIST ",count,node,net); for(i=1;i<count;i++)
printf("%d,",core[i]); printf("%d %s\n",core[count],prog); node=$1; count=1;
core[count]=$2}; END {printf("-n %d -host %s -env %s -env I_MPI_PIN_PROCESSOR_LIST
",count,node,net); for(i=1;i<count;i++) printf("%d,",core[i]);
printf("%d %s\n",core[count],prog))}' >rankfile.${id}

mpirun -r ssh -f $PBS_NODEFILE -configfile rankfile.${id} 2>&1 | tee -a intelmpi.log

```

```
echo end time is `date` | tee -a intelmpi.log
```

2 附录—Linux 常用命令

1.1 浏览目录命令

用户使用命令行所做的大部分工作是用来定位、列出、创建以及删除文件和目录，下面列举最为常用的这类命令及其解释，更为详细的用法请参见 Linux 有关书籍。

➤ `ls [options] [directory]` 列出文件

常用的命令参数选项有 `-l`，`-a`，`-t` 等。`ls` 代表 `list`。

`ls -la` — 给出当前目录下所有文件的一个长列表，包括以句点开头的隐藏文件。

`ls -l *.doc` — 列出当前目录下以字母 `.doc` 结尾的所有文件。

`ls -a` — 显示当前目录所有文件及目录。

`ls -d` — 将目录像文件一样显示，而不显示该目录下的文件。

`ls -R` — 列出所有子目录下的文件。

`ls -t` — 将文件依建立时间之先后次序列出。

`ls -ltr s*` — 列当前目录下任何名称是 `s` 开头的文件，愈新的文件排愈后。

➤ `cd [directory]` 切换目录

`cd` 代表 `change directory`。

`cd ~` — 切换到用户家目录。

`cd /tmp` — 切换到目录 `/tmp`。

`cd ..` — 切换到上一层目录

`cd /` — 切换到系统根目录

`cd /usr/bin` — 切换到 `/usr/bin` 目录。

1.2 浏览文件命令

➤ `cat [textfile]` 显示文本文件内容

`cat` 代表 `catenate`。

`cat /etc/passwd` — 显示文本文件 `passwd` 中的内容。

`cat test.txt | more` — 逐页显示 `test.txt` 文件中的内容。

`cat test.txt >>test1.txt` — 将 `test.txt` 的内容附加到 `test1.txt` 文件之后。

`cat a.txt b.txt >readme.txt` — 将文件 `a.txt` 和 `b.txt` 合并成 `readme.txt` 文件。

➤ `more [textfile]` 和 `less [textfile]` 逐屏显示文本文件内容

`more` 命令和 `less` 命令都是用于要显示的内容会超过一个画面长度的情况。`more` 命令让画面在显示满一页时暂停，此时可按空格键继续显示下一个画面；而 `less` 命令除了可以按空格键向下显示文件外，还可以利用上下键来卷动文件。二者都使用热键 `q` 退出。

`more /etc/passwd` — 显示 `etc` 目录下文本文件 `passwd` 中的内容。

`ls -al | more` — 以长格形式显示当前目录下的所有内容，显示满一个画面便暂停，可按空格键继续显示下一画面。按热键 `q` 退出。

`less /etc/named.conf` — 显示 `etc` 目录下文本文件 `named.conf` 中的内容。

`ls -al | less` — 以长格形式显示当前目录下的所有内容，用户可按上下键浏览。按热键 `q` 退出。

➤ `head [files]` 和 `tail [files]` 查看文件前几行和后几行的内容

`head` 和 `tail` 命令用于查看从文件头或文件尾开始的指定数量的行的内容。

`head -10 /etc/passwd` — 显示 `/etc/passwd` 文件的前 10 行内容。

`tail -10 /etc/passwd` — 显示 `/etc/passwd` 文件的倒数 10 行内容。

`tail +10 /etc/passwd` — 显示 `/etc/passwd` 文件的从第 10 行开始到末尾的内容。

`head -20 file | tail -10 /etc/passwd` — 结合 `head` 与 `tail` 命令，显示 `/etc/passwd` 文件的第 11 行到第 20 行的内容。

`tail -f /usr/tmp/logs/daemon_log.txt` — 使用参数 `-f` 时，`tail` 不会回传结束信号，除非我们去自行去中断它；相反地，它会一直不停地继续显示，直到发现文件自它最后一次被读取后，又被加入新的内容时。一般用于监视日志文件的动态更新，有实时监视的效果。本例用于显示 `/usr/tmp/logs/daemon_log.txt` 文件的动态更新。

1.3 目录操作命令

➤ `pwd` 显示用户目前所在的工作目录的绝对路径名称。

`pwd` 代表 `print working directory`

➤ `mkdir [-p] [directory]` 创建目录

`mkdir` 代表 `make directory`。

`mkdir mydir` — 在当前目录下建立 `mydir` 目录。

`mkdir -p one/two/three` — 在当前目录下建立指定的嵌套子目录。

➤ `rmdir [-p] [directory]` 删除目录

删除“空”的子目录。`rmdir` 代表 `remove directory`。

`rmdir mydir` — 删除“空”的子目录 `mydir`。

`rmdir -p one/two/three` — 删除“空”的嵌套子目录 `one/two/three`。

注意：选项“-p”表示可以递归删除多层子目录，但删除的目录须为空目录，且须具有对该目录的写入权限。

1.4 文件操作命令

➤ `cp [source] [target]` 复制文件

将一个文件、多个文件或目录复制到另一个地方。`cp` 代表 `copy`。

`cp test1 test2` — 将文件 `test1` 复制成新文件 `test2`。

`cp test3 /home/bible/` — 将文件 `test3` 从当前目录复制到 `/home/bible/` 目录中。

`cp -r dir1(目录) dir2(目录)` — 复制目录 `dir1` 为目录 `dir2`。`-r` 参数表示递归。

注意：`cp` 命令默认将覆盖已存在的文件，加 `-i` 参数表示覆盖前将与用户交互。

➤ `mv [source] [target]` 移动文件，文件改名

将文件及目录移到另一目录下，或更改文件及目录的名称。`mv` 代表 `move`。

`mv afile bfile` — 将文件 `afile` 改名成新文件 `bfile`。

`mv afile /tmp` — 将文件 `afile` 从当前目录移动到 `/tmp/` 目录下。

`mv afile ../` — 将文件 `afile` 移动到上层目录。

`mv dir1 ../` — 将目录 `dir1` 移动到上层目录。

➤ `rm [files]` 删除文件或目录

删除目录需要加 `-r` 选项，强制删除用 `-f`。`rm` 代表 `remove`。

`rm myfiles` — 删除 `myfiles` 文件。

`rm *` — 删除当前目录下的所有未隐藏文件。

`rm -f *.txt` — 强制删除所有以后缀名为 `txt` 文件。

`rm -rf mydir` — 删除目录 `mydir` 以及其下的所有内容。

`rm -ia*` — 删除当前目录下所有以字母 `a` 开头的文件，`-i` 选项表示将与用户交互。

➤ `ln [-s] [source] [target]` 建立链接

在文件和目录之间建立链接，参数 `-s` 为建立软链接（符号链接）。`ln` 代表 `link`。

`ln -s /usr/share/doc doc` — 创建链接文件 `doc`，并指向目录 `/usr/share/doc`。

`ln -s afile linkafile` — 为文件 `afile` 创建名为 `linkafile` 的软链接

`ln afile bfile` — 为文件 `afile` 创建名为 `bfile` 的硬链接

`ln /usr/share/test hard` — 创建一个硬链接文件 `hard`，这时对于 `test` 文件对应的存储区域来说，又多了一个文件指向它。

➤ `touch [options] [filename]` 新建一个文本文件

新建一个文本文件或修改文件的存取/修改的时间记录值。

`touch *` — 将当前目录下的文件时间修改为系统的当前时间。

`touch -d 20100101 test` — 将 `test` 文件的日期改为 2010 年 1 月 1 日。

`touch abc` — 若 `abc` 文件存在，则修改为系统的当前时间；若不存在，则生成一个为当前时间的空文件。

➤ `file [filename]` 查看 `filename` 文件的类型

1.5 查找类命令

➤ `grep 'string' [file]` 在文件中搜索匹配的字符串位置（所在行）并输出到屏幕

`grep` 代表（global regular expression print，全局正则表达式打印）。

`grep bible /etc/exports` — 查找文件 `/etc/exports` 中包含字符串 `bible` 的所有行。

`grep -v ^# /etc/apache2/httpd.conf` — 在主 Apache 配置文件中，查找所有非注释行。

`tail -100 /var/log/apache/access.log | grep 404` — 在 Web 服务器日志的后 100 行中查找包含字符串 `404` 的行，`404` 代表 Web 服务器的“文件没找到”代码。

`tail -100 /var/log/apache/access.log | grep -v googlebot` — 在 Web 服务器的后 100 行中，查看没有被 Google 搜索引擎访问的行。

`rpm -qa | grep httpd` — 搜索已安装的 rpm 包中含有 `httpd` 字符串的文件名。

➤ `find name [filename]` 和 `locate [file]` 查找文件或目录

`find` 用来查找文件或目录。`locate` 用于快速查找定位文件，但只能搜索文件名。

`find ./ -name httpd.conf` — 搜索当前目录下名为 `httpd.conf` 的文件并显示结果。

`find /etc -name httpd.conf` — 搜索 `/etc` 目录下名为 `httpd.conf` 的文件并显示结果。

`find . | grep page` — 在当前目录及其子目录中，查找文件名包含字符串 `page` 的文件。

`locate traceroute` — 在系统任何地方查找文件名包含字符串 `traceroute` 的文件。

➤ `whereis [options]` 查找程序的源、二进制文件或手册

`whereis` 命令在指定的目录中查找程序的源、二进制文件或手册。

`whereis passwd` — 将和 `passwd` 文件相关的文件都查找出来。

`whereis -b passwd` — 只将二进制文件查找出来。

1.6 用法帮助命令

➤ `man [command]` 查看 `command` 命令的说明文档

`man` 代表 `manual page`

➤ `[command] -h` 或 `-help, --h, --help`

查看 `command` 命令的说明文档

➤ `info [command]` 查看 `command` 命令的说明文档

`info` 代表 `information`

➤ `whatis [command]` 在 `whatis` 资料库（手册）中搜寻指定命令的简短描述。

1.7 打包、解包，压缩、解压缩命令

➤ `tar [options] [filename]` 打包命令。

`tar` 代表 `tape archive`。它能够将用户所指定的文件或目录打包成一个文件，但不做压缩。一般 Linux/Unix 上常将打包命令 `tar` 与压缩 `gzip` 联合使用。`Tar` 不仅可以打包文件，也可以将硬盘数据备份。`tar` 命令常用参数：

`-c`: 创建一个新 `tar` 文件

`-v`: 显示运行过程的信息

`-f`: 指定文件名

`-z`: 调用 `gzip` 压缩命令进行压缩或解压

`-j`: 调用 `bzip2` 压缩命令进行压缩或解压

-t: 查看压缩文件的内容

-x: 解开 tar 文件

-p: 使用原文件的原来属性（属性不会依据使用者而变）

`tar -cvf test.tar *` — 将所有文件打包成 `test.tar`, 扩展名 `.tar` 需自行加上。

`tar -zcvf test.tar.gz *` — 将所有文件打包并调用 `gzip` 命令压缩成为 `test.tar.gz`。

`tar -tf test.tar` — 查看 `test.tar` 文件中包括了哪些文件。

`tar -xvf test.tar` — 将 `test.tar` 文件解开。

`tar -zxvf foo.tar.gz` — 将 `foo.tar.gz` 解压缩。

`tar -jxvf foo.tar.bz2` — 将 `foo.tar.bz2` 解压缩。

`tar -cvf /tmp/etc.tar /etc` — 将整个 `/etc` 目录下的文件全部打包成为 `/tmp/etc.tar`。

`tar -zcvf /tmp/etc.tar.gz /etc` — 将整个 `/etc` 目录下的文件全部打包并调用 `gzip` 命令压缩成为 `/tmp/etc.tar.gz`。

`tar -zxvpf /tmp/etc.tar.gz /etc` — 将 `/etc/` 内的所有文件备份下来, 并且保存其权限。参数 `-p` 非常重要, 尤其是当需要保留原文件的属性时!

➤ `gzip [options] [filename]` 压缩和解压缩命令。

通过压缩减少文件大小有两个明显的好处, 一是可以减少存储空间, 二是通过网络传输文件时, 可以减少传输的时间。`gzip` 和 `gunzip` 是在 Linux 系统中经常使用的一个对文件进行压缩和解压缩的命令。

`gzip` 代表 GNU zip。GNU 是 Gnu is Not Unix 的缩写, GNU

Project 是自由软件基金会 (Free Software Foundation) 的一部分, 它对 Linux 下的许多编程工具负责。

各选项的含义:

-c: 将压缩结果写入到标准输出上, 原文件保持不变。缺省时 `gzip` 将原文件压缩为 `.gz` 文件, 并删除原文件。

-r: 递归式地查找指定目录并压缩其中的所有文件或者是解压缩。

-d: 解压缩指定文件。

-t: 测试压缩文件的完整性。

-v: 对每一个压缩和解压的文件, 显示文件名和压缩比。

`gzip usr.tar` — 压缩一个文件 `usr.tar`, 此时压缩文件的扩展名为 `.tar.gz`。

`gzip -v /mnt/lgx/a1.doc` — 压缩文件/mnt/lgx/a1.doc，此时压缩文件的扩展名为.gz。

`gzip -d /mnt/lgx/a1.doc.gz` — 解压缩文件/mnt/lgx/a1.doc.gz

1.8 时间相关命令

➤ `date` 显示/修改当前的系统时间

`date` — 查看系统当前时间。

`date 121010232009.10` — 将时间更改为 12 月 10 日 10 点 23 分 10 秒 2009 年[月日時分年.秒]。

➤ `cal` 显示日历

`cal` — 显示当月日历。

`cal 7 2007` — 显示 2007 年 7 月份的日历。

`cal 2010` — 显示 2010 年全年的日历。

➤ `hwclock` 显示当前的硬件时钟

`hwclock --show` — 查看硬件当前时钟。

`hwclock --set --date="01/17/2010 13:26:00"` — 设置硬件时钟，格式 `hwclock --set --date="月/日/年 时：分：秒"`。

`hwclock --hctosys` — 硬件时钟与系统时间同步。`--hctosys` 表示 Hardware Clock to SYStem clock。

`hwclock --systohc` — 系统时间和硬件时钟同步。

➤ `ntpdate` 同步网络时钟

`ntpdate 210.72.145.44` — 与 ntp 时间服务器进行时间同步。210.72.145.44 是中国国家授时中心的官方服务器。（需要安装 ntp 的软件包）。

1.9 系统信息类命令

➤ `dmesg` 显示系统开机信息命令

`dmesg` 代表 diagnostic message。显示系统诊断信息、操作系统版本号、物理内存的大小以及其它信息。

- `df` 用于查看文件系统的各个分区的占用情况。`df` 代表 `disk free`。
`df -hl` — 查看磁盘剩余空间信息。
`df -T` — 显示分区类型。
- `fdisk` 磁盘分区工具
`fdisk -l` — 显示所有硬盘的分区情况。
- `du [options] [directory or filename...]` 显示指定的目录或文件所占用的磁盘空间。`du` 代表 `disk usage`
- `free` 查看系统内存，虚拟内存（交换空间）的大小占用情况
- `who` 或 `w` 查看当前系统中有哪些用户登录
`who` — 显示登录的用户名、登录终端和登录时间。
`who -uH` — 带有标题栏的登录用户的详情，其中 `-u` 选项指定显示用户空闲时间。

1.10 网络通讯类命令

- `ifconfig` 显示和设置网络设备
`ifconfig eth0 192.168.0.1` — 将第一块网卡的 IP 地址设置为 `192.168.0.1`。
`ifconfig eth0 down` — 关闭第一块网卡。
`ifconfig eth0 up` — 启用第一块网卡。
`ifconfig eth0 netmask 255.255.255.0` — 将第一块网卡的子网掩码设置为 `255.255.255.0`。
`ifconfig eth0 192.168.0.1 netmask 255.255.255.0` — 同时设置 IP 地址和子网掩码。
`ifconfig eth0 -broadcast 192.168.0.255` — 将第一块网卡的广播地址设置为 `192.168.0.255`。
- `route` 显示和设置路由
`route add 0.0.0.0 gw 网关地址` — 增加一个默认路由。
`route del 0.0.0.0 gw 网关地址` — 删除一个默认路由。
`route` — 显示当前路由表。
- `ping [options] [主机名/IP 地址]`，检测是否能够与远端机器建立网络通讯连接

- `netstat [options]` 查看网络状态

`netstat -i` —interface, 显示网络界面信息表单。

`netstat -s` —statistic, 显示网络工作信息统计表。

`netstat -t` —tcp, 显示 TCP 传输协议的连接状态。

`netstat -r` —route, 显示路由表。

- `traceroute [远程主机 IP 地址或域名]` 跟踪路由

- `ftp` 文件传输

- `telnet [主机名/IP 地址]` 登录到远程计算机

- `finger` 查询远程计算机 (通常是运行 Linux/UNIX 的计算机) 上用户的详细信息。

1.11 软件包管理命令

RPM 的全名是 Red Hat Package Manager。利用 RPM 命令, 可以安装、删除、升级管理软件, 支持在线安装和升级软件。通过 RPM 包管理可以知道软件包包含哪些文件, 也可以查询系统中的某个文件属于哪个软件包, 可以查询系统中的软件包是否安装及安装的版本。具体用法请参见 Linux 相关书籍。

下面列举一些 rpm 的基本用法。

- 我们得到一个新软件, 在安装之前, 一般都要先查看一下这个软件包是做什么的, 可以用这条命令查看:

```
rpm -qpi strace-4.5.18-10.13.x86_64.rpm
```

系统将会列出这个软件包的详细资料。

- 我们可以用下面这条命令查看软件包将会在系统里安装哪些文件:

```
rpm -qpl strace-4.5.18-10.13.x86_64.rpm
```

- 安装该软件包:

```
rpm -ivh strace-4.5.18-10.13.x86_64.rpm
```

- 如果系统已经安装该软件包的低版本, 可以用下面的命令进行升级安装:

```
rpm -Uvh strace-4.5.18-10.13.x86_64.rpm
```

安装某个软件时，RPM会自动处理包的依赖关系，如果不想进行依赖检查，可以给rpm加上 `--nodeps` 参数，想要强制安装可以加上 `--force` 参数。

- 卸载某个安装过的软件，只需执行 `rpm -e <文件名>` 命令即可。

```
rpm -e strace
```

- 如果不小心误删了某些包的系统文件，可以用下列命令查看有哪些文件损坏，以便进行修复安装。

```
rpm -Va
```

- 下面这条命令行可以帮助我们快速判定某个文件是属于哪个软件包：

```
rpm -qf <文件名>
```

- 如果想查看当前系统已经安装了哪些rpm包，可以执行：

```
rpm -qa
```

也可以与 `grep` 联用，进行查找操作：

```
rpm -qa | grep strace
```

1.12 编辑器命令 (vim)

在 Linux 下编写文本或语言程序，首先必须选择一种文本编辑器。VIM 编辑器是工作在字符模式下的高效率文本编辑器，它可以执行输出、删除、查找、替换、块操作等众多文本操作，而且用户可以根据自己的需要对其进行定制。

在命令行里输入 `vim` 即可调用 VIM 编辑器：

`vim` — 调用 `vim`，可以进行编辑工作，编辑完成后可以保存到新文件；

`vim foo.txt` — 如果文件已经存在，则打开编辑，如果文件不存在，则新建编辑；

VIM 有几种基本工作模式，在 VIM 里头执行 `:help mode` 可以看到 VIM 的所有模式，主要有如下几种。

- Normal mode 即通常所谓的命令模式，在此模式使用 `a`、`i`、`A`、`I`、`o`、`O` 等进入 Insert mode。
- Insert mode 即通常所谓的编辑模式，在此模式使用 `ESC` 进入 Normal mode 或者 `Ctrl-o` 临时进入 Normal mode。
- Command-line mode 命令行模式，在 Normal mode 下按冒号进入，按 `ESC` 取消执行命令或者回车执行命令，然后回到 Normal mode。
- Visual mode 即选择模式（注意跟用鼠标选择不同），用 `v`、`V`、`C-v` 或 `C-q` 进入

- `Select mode` 鼠标选择
- `Replace mode` 在`Normal mode`下按`R`进入，按`ESC`返回`Normal mode`，相当于Windows下命令行中按`Insert`键进入的覆盖模式

从 **VI/VIM** 中退出：按 `ESC` 确认返回到 `Normal mode`，然后

- `:wq` 保存并退出
- `:q!` 不保存并退出
- `:x` 退出，如果文件更改则保存
- `ZZ` 退出，如果文件更改则保存（按住`Shift`，再按两次`z`）

VIM 的命令是非常具有美感的，下面是几个例子：

- `i` 在光标前插入；`I` 在行首插入
- `a` 在光标后插入；`A` 在行末插入
- `o` 在下一行插入；`O` 在上一行插入
- `x` 删除当前字符；`X` 删除前一个字符

类似的还有 `b`，`B`，`ge`，`gE`，`w`，`W`，`e`，`E`，`f`，`F`，`t`，`T`，这些命令都可以用 `:help cmdname` 查到帮助。

- `dd` 删除一行
- `yy` 拷贝一行
- `h j k l` 左下上右移动光标
- `Ctrl-w h`，`Ctrl-w j`，`Ctrl-w k`，`Ctrl-w l` 切换到左下上右窗口

VI/VIM的命令大多都可以带一个数字前缀或者一个数字范围，比如：

- `5dd` 从当前行开始删除5行
- `5yy` 从当前行开始拷贝5行
- `1,3d` 删除1至3行
- `1,3y` 拷贝1至3行（`.`代表当前行，`$`代表最后一行）

VIM 的功能非常强大，想进一步了解和学习的 **VIM**，可以参考 **VIM** 相关文档。

1.13 用户管理命令

`useradd` 添加用户

`userdel` 删除用户

`passwd` 为用户设置密码

`usermod` 修改用户的登录名、用户的家目录等

`id` 查看用户的 `UID`、`GID` 及所归属的用户组

`pwck` 校验用户配置文件 `/etc/passwd` 和 `/etc/shadow` 文件内容是否合法或完整

`chfn` 更改用户信息工具。可以留下真实的姓名、办公室、电话等资料。

`su` 用户切换工具。表示 `substitute user`

`sudo` 通过另一个用户来执行命令 (`execute a command as another user`)

`finger` 查看用户信息工具

1.14 用户组管理命令

`groupadd` 添加用户组

`groupdel` 删除用户组

`groupmod` 修改用户组信息

`groups` 显示用户所属的组

`grpck` 校验组账号文件 (`/etc/group`) 和影子文件 (`/etc/gshadow`) 的一致性和正确性。

1.15 输入/输出重定向与管道命令

➤ `[command]<inputfile` 输入重定向

输入重定向。输入重定向是指把命令（或可执行程序）的标准输入重定向到指定的文件中。也就是说，输入可以不来自键盘，而来自一个指定的文件。所以说，输入重定向主要用于改变一个命令的输入源，特别是改变那些需要大量输入的输入源。

➤ `[command]>outputfile` 输出重定向

输出重定向是指把命令（或可执行程序）的标准输出或标准错误输出重新定向到指定文件中。这样，该命令的输出就不显示在屏幕上，而是写入到指定文件中。

`ls -lR >dirtree.list` — 创建一个包含目录树列表的文件。

➤ `[command] >>outputfile` 输出追加重定向

为避免输出重定向中指定文件只能存放当前命令的输出重定向的内容，`shell` 提供了输出重定向的一种追加手段。输出追加重定向与输出重定向的功能非常相似，区别仅在于输出追加重定向的功能是把命令（或可执行程序）的输出结果追加到指定文件的最后，而该文件原有内容不被破坏。如果文件不存在，那么就创建它，如果存在，那么就追加到文件后边。

➤ `[command1] | [command2]` 把 `command1` 执行的结果作为输入送到 `command2` 中执行。

