Introduction

In this workshop we shall give you hands-on experience preparing your own computer for a local area network (LAN), logging in and transferring files between computers, administering your own and other computers on your LAN or on the internet, and uploading or downloading stuff from your home computer or from anywhere.

GNU/Linux programs we are going to use

ifconfig, route, arp-scan: to manipulate IP addresses and interfaces;

sshd: to recognise a login attempt, accept it and deal with it;

ssh: to log into a remote machine and for executing commands;

screen: to share interactive command-line sessions between computers;

scp: to copy files between computers on a network using encryption, authentication and security.

sftp: to transfer files to and from a remote computer.

Exercises we shall do, using these programs

1. **set up local area network** and ensure that everything works.

2. **access your computer** by allowing incoming logins;

3. **conduct remote administration** by logging into remote computers;

4. **create interactive 'help' session** for teaching commands.

5. **manage remote file transfers** from/to you and between remote computers.

colour prompt: Note this extremely useful addition that makes a colour prompt when you login using `ssh`, so you do not do something stupid thinking you are on your own computer. Add this bit to a computer account under your control by running this on the computer:

```
$ cd
$ cat >> .bashrc < eof
# if got here by ssh login, set colour prompt to remind us
if [ -n "$SSH_CLIENT" ]; then text=" ssh"
export PS1='\[\e[0;31m\]\u@\h:\w${text}$\[\e[m\] '
fi
eof
```

Caution: *O'Toole's Law* [1] is everywhere rampant. In the end, we have to get stuff working. When things go wrong, we'll need diagnostic tools; we'll pick them up as we go ...

[1] Murphy was an optimist

1 Hands-On Exercise: Create an Interactive 'help' Session

In the old days, we used `rlogin` for remote login and `rsh` for remote command execution. The need for secure communication over untrusted channels means that we now have to use the secure versions `slogin` and `ssh` instead. As a first step in dealing with other computers, I shall get you to log in to the `lsga` computer and then log back into your computer. This way, all your commands are run from your computer but monitored from mine, and so I can interactively help to correct errors without walking around to each computer. For this, we create a Local Area Network and assign all computers an IP address, make a shared `screen` session and monitor the command-line.

Let's do it!

1.1 Configure your computer to use the 'sudo' command

```
$ su -c "/usr/sbin/visudo"  [and seek help from us if necessary]
```

1.2 Install and configure the 'secure shell' suite

```
[others] install openssh-client and openssh-server from your repositories
[debian] $ sudo apt-get install openssh-client openssh-server
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  ssh-askpass rssh molly-guard ufw monkeysphere
The following NEW packages will be installed:
  openssh-server
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 0 B/343 kB of archives.
After this operation, 828 kB of additional disk space will be used.
Preconfiguring packages ...
Selecting previously unselected package openssh-server.
(Reading database ... 142228 files and directories currently installed.)
Unpacking openssh-server (from .../openssh-server_1%3a6.0p1-4+deb7u2_i386.deb)
Processing triggers for man-db ...
Setting up openssh-server (1:6.0p1-4+deb7u2) ...
Creating SSH2 RSA key; this may take some time ...
Creating SSH2 DSA key; this may take some time ...
Creating SSH2 ECDSA key; this may take some time ...
[ ok ] Restarting OpenBSD Secure Shell server: sshd.

### check that the secure-shell daemon started and is running ###
$ sudo which sshd
/usr/sbin/sshd
```

1.3 Stop the Network-Manager from 'managing' your network

```
Each distro. has its own peculiar way of doing this.
$ sudo /etc/init.d/network-manager stop       [Debian]
$ sudo service network-manager stop           [Debian]
$ sudo stop network-manager                   [Ubunutu, Mint]
$ sudo systemctl stop NetworkManager.service  [Fedora]
$ sudo service NetworkManager stop            [CentOs]
```

1.4 Establish your unique IP address for this workshop

```
### make interface (e.g. 'eth0') unavailable whilst configuring it ###
$ sudo /sbin/ifconfig eth0 down

### give it an IP address and make it available for routing ###
$ sudo /sbin/ifconfig eth0 192.168.1.44 up        [e.g. IP=192.168.1.44]

### check that it all worked ###
$ /sbin/ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:02:3f:b8:1c:38
          inet addr:192.168.1.44  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::202:3fff:feb8:1c38/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:690 errors:0 dropped:0 overruns:0 frame:0
          TX packets:681 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:56124 (54.8 KiB)  TX bytes:55841 (54.5 KiB)
          Interrupt:19 Base address:0x3000

### check that you can route through the eth0 interface ###
$ /sbin/route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.1.0     0.0.0.0         255.255.255.0   U     0      0        0 eth0
```

1.5 See who else is on your Local Area Network

```
### discover any other IP addresses accessable through eth0 ###
### (sometimes need to run this 2 or 3 times to see all computers) ###
$ sudo /usr/bin/arp-scan --interface=eth0 192.168.1.1/24
Interface: eth0, datalink type: EN10MB (Ethernet)
WARNING: host part of 192.168.1.1/24 is non-zero
Starting arp-scan 1.8.1 with 256 hosts
192.168.1.22 00:02:3f:b1:30:69 Compal Electronics, Inc.
```

```
192.168.1.33 00:02:3f:b8:1c:38 Compal Electronics, Inc.
192.168.1.250 00:60:64:91:83:a5 NETCOMM LIMITED
192.168.1.100 00:14:c2:d8:67:c0 Hewlett Packard
4 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.8.1: 256 hosts scanned in 1.327 seconds. 4 responded

### check if the computer you want to reach is responding to requests ###
$ ping -c 1 192.168.1.100
PING 192.168.1.100 (192.168.1.100) 56(84) bytes of data.
64 bytes from 192.168.1.100: icmp_req=1 ttl=64 time=0.663 ms
--- 192.168.1.100 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.663/0.663/0.663/0.000 ms

### keep going until all machines on your LAN are found. ###
### check cables, network-manager, /etc/ssh/sshd_config and firewall if problems.
```

1.6 Ensure that your computer can be accessed using 'slogin'

Nothing useful happens until a computer on your LAN can recognise and accept your login attempt. Thus it is of prime importance to set up `openssh-server` with its daemon (`sshd`) which looks for and deals with your login.

```
### see if the ssh daemon program (for taking incoming login attempts) is running
$ sudo which sshd

### no response means 'openssh-server' package is not installed. Install it ###

### check now if the ssh daemon is present ###
$ sudo which sshd
/usr/sbin/sshd

### check if the ssh daemon is actually running ###
$ pidof sshd
3216               [some integer should appear here if it is running]

### if sshd is not running, force it to start ###
$ sudo /etc/init.d/ssh restart      [debian kubuntu]
$ sudo rcsshd restart               [suse]
$ sudo /etc/init.d/sshd restart     [centos fedora redhat]
$ sudo /etc/rc.d/sshd restart       [freeBSD archlinux]
$ sudo /etc/rc.d/rc.sshd restart    [slackware]
Restarting OpenBSD Secure Shell server: sshd.
```

1.7 Log in to computer ‘lsga’ using ‘slogin’

Now `sshd` is available to handle a login. First we shall log in to computer `lsga` using `slogin` (which is linked to `ssh` in most distros). While there, you can attach to the shared `screen` session. Finally you return to your computer using `slogin`, from whence you conduct the rest of the exercises.

```
### find out where the secure login program is ###
$ /bin/ls -l $(which slogin)
lrwxrwxrwx 1 root root 3 Apr  3 09:42 /usr/bin/slogin -> ssh

### use ssh to log in to computer ‘lsga’ for the first time ###
$ ssh lsga@192.168.1.100
The authenticity of host ‘192.168.1.100 (192.168.1.100)’ can’t be established.
ECDSA key fingerprint is 54:6c:00:85:4b:30:4f:c5:12:68:b4:e7:7c:4e:0b:c1.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added ‘192.168.1.100’ (ECDSA) to the list of known hosts.
lsga@192.168.1.100’s password:
Linux lsga 3.2.0-4-686-pae #1 SMP Debian 3.2.60-1+deb7u1 i686
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.

# flush your firewall temporarily to allow ssh in (if it is causing problems)
$ sudo /sbin/iptables -F
```

1.8 On ‘lsga’, a screen session has been started before this

```
$ screen -S 44   [On lsga: a screen session made for you named 44, for example]
```

1.9 Attach to your screen session so we can share the command line

```
$ screen -x 44   [connect up with screen session 44; I’ll tell you which]
```

1.10 Now log back in to your computer to do all the exercises

```
$ ssh you@192.168.1.44         [if you established that as your IP address]
```

1.11 Increase the amount of debugging information if needed

```
## can use -v -vv or -vvv to get more and more info. about the login ###
$ ssh -v user@snowy.esd.net
OpenSSH_6.0p1 Debian-4+deb7u1, OpenSSL 1.0.1e 11 Feb 2013
debug1: Reading configuration data /etc/ssh/ssh_config
debug1: /etc/ssh/ssh_config line 19: Applying options for *
debug1: Connecting to snowy.esd.net [192.168.1.20] port 22.
```

```
debug1: Connection established.
debug1: identity file /home/user/.ssh/id_rsa type 1
debug1: Checking blacklist file /usr/share/ssh/blacklist.RSA-2048
debug1: Checking blacklist file /etc/ssh/blacklist.RSA-2048
debug1: identity file /home/user/.ssh/id_rsa-cert type -1
debug1: identity file /home/user/.ssh/id_dsa type 2
debug1: Checking blacklist file /usr/share/ssh/blacklist.DSA-1024
debug1: Checking blacklist file /etc/ssh/blacklist.DSA-1024
debug1: identity file /home/user/.ssh/id_dsa-cert type -1
debug1: identity file /home/user/.ssh/id_ecdsa type -1
debug1: identity file /home/user/.ssh/id_ecdsa-cert type -1
debug1: Remote protocol version 2.0, remote software version OpenSSH_5.2
debug1: match: OpenSSH_5.2 pat OpenSSH_5*
debug1: Enabling compatibility mode for protocol 2.0
debug1: Local version string SSH-2.0-OpenSSH_6.0p1 Debian-4+deb7u1
debug1: SSH2_MSG_KEXINIT sent
debug1: SSH2_MSG_KEXINIT received
debug1: kex: server->client aes128-ctr hmac-md5 none
debug1: kex: client->server aes128-ctr hmac-md5 none
debug1: SSH2_MSG_KEX_DH_GEX_REQUEST(1024<1024<8192) sent
debug1: expecting SSH2_MSG_KEX_DH_GEX_GROUP
debug1: SSH2_MSG_KEX_DH_GEX_INIT sent
debug1: expecting SSH2_MSG_KEX_DH_GEX_REPLY
debug1: Server host key: RSA 7f:1b:85:cd:65:bc:4c:e8:ef:39:2a:3a:91:ab:b5:b2
debug1: Host ‘snowy.esd.net’ is known and matches the RSA host key.
debug1: Found key in /home/user/.ssh/known_hosts:18
debug1: ssh_rsa_verify: signature correct
debug1: SSH2_MSG_NEWKEYS sent
debug1: expecting SSH2_MSG_NEWKEYS
debug1: SSH2_MSG_NEWKEYS received
debug1: Roaming not allowed by server
debug1: SSH2_MSG_SERVICE_REQUEST sent
debug1: SSH2_MSG_SERVICE_ACCEPT received
debug1: Authentications that can continue: publickey,keyboard-interactive
debug1: Next authentication method: publickey
debug1: Offering RSA public key: /home/user/.ssh/id_rsa
debug1: Authentications that can continue: publickey,keyboard-interactive
debug1: Offering DSA public key: /home/user/.ssh/id_dsa
debug1: Authentications that can continue: publickey,keyboard-interactive
debug1: Trying private key: /home/user/.ssh/id_ecdsa
debug1: Next authentication method: keyboard-interactive
Password:
```

2 Hands-On Exercise: Conduct Remote Administration

Log in to an account on the remote computer and execute commands while there.

2.1 Login to a remote computer

```
$ remote=192.168.1.100
$ ssh user@$remote
The authenticity of host ‘192.168.1.100 (192.168.1.100)’ can’t be established.
ECDSA key fingerprint is 54:6c:00:85:4b:30:4f:c5:12:68:b4:e7:7c:4e:0b:c1.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added ‘192.168.1.100’ (ECDSA) to the list of known hosts.
user@192.168.1.100’s password:
Linux remote 3.2.0-4-686-pae #1 SMP Debian 3.2.60-1+deb7u1 i686
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
user@remote:~ ssh$ exit
```

2.2 Configure automatic (passwordless) login

```
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/user/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/user/.ssh/id_rsa.
Your public key has been saved in /home/user/.ssh/id_rsa.pub.
The key fingerprint is:
16:b1:f6:37:ef:90:4c:be:9b:41:fa:5a:80:f0:5e:4a user@brahma
The key's randomart image is:
+--[ RSA 2048]----+
|        .        |
|         o       |
|      . +        |
|       + +       |
|        E + =    |
|       + o O +   |
|        o . B .  |
|           o *   |
|          ..=..  |
+-----------------+

### copy this id to the correct place on the remote m/c ###
```

```
$ ssh-copy-id user@$remote
user@remote's password:
Now try logging into the machine, with "ssh 'lsga@192.168.1.100'", and check in:
  ~/.ssh/authorized_keys
to make sure we haven't added extra keys that you weren't expecting.

$ ssh user@$remote
Linux remote 3.2.0-4-686-pae #1 SMP Debian 3.2.60-1+deb7u1 i686
user@remote:~ ssh$
```

2.3 Execute one command on remote computer

```
$ /bin/ls -l $(which ssh)
-rwxr-xr-x 1 root root 440816 Apr  3 09:42 /usr/bin/ssh

### specify a compound command within " " quotes ###

$ ssh lsga@192.168.1.100 "mkdir -p temp; touch temp/a; /bin/df -h"
Filesystem      Size  Used Avail Use% Mounted on
rootfs           27G  6.1G   19G  25% /
udev             10M     0   10M   0% /dev
tmpfs            74M  676K   74M   1% /run
/dev/sda1        27G  6.1G   19G  25% /
tmpfs           5.0M     0  5.0M   0% /run/lock
tmpfs           390M   80K  390M   1% /run/shm
```

3 Hands-On Exercise: Manage Remote File Transfer

Copy files and folders between remote computers and to or from your computer.

```
## from or to your computer ###
$ scp file lsga@192.168.1.100:~/newfile
$ ls -l file
-rw-r--r-- 1 lsga lsga 0 2014-07-14 14:54 file
$ scp file lsga@192.168.1.100:~/newfile
file                                         100%    0     0.0KB/s   00:00

### between two third parties ###
$ scp -3 lsga@192.168.1.100:~/temp/file lsga@192.168.1.33:~/temp/newfile
```

You might think that sftp is the same as scp but it is not! To use `scp` you need to know where all the files are. If you don't, then `sftp` is the one to find out for you.

```
$ sftp localhost
lsga@localhost's password:
Connected to localhost.
sftp> version
SFTP protocol version 3
sftp> exit

$ sftp lsg@192.168.1.100
lsga@192.168.1.100's password:
Connected to 192.168.1.100.
sftp> help
Available commands:
bye or quit or exit                 Quit sftp
cd path                             Change remote directory to 'path'
chgrp grp path                      Change group of file 'path' to 'grp'
chmod mode path                     Change permissions of file 'path' to 'mode'
chown own path                      Change owner of file 'path' to 'own'
df [-hi] [path]                     Display statistics for current directory
get [-Ppr] remote [local]           Download file
help                                Display this help text
lcd path                            Change local directory to 'path'
lls [ls-options [path]]             Display local directory listing
lmkdir path                         Create local directory
ln [-s] oldpath newpath             Link remote file (-s for symlink)
lpwd                                Print local working directory
ls [-1afhlnrSt] [path]              Display remote directory listing
lumask umask                        Set local umask to 'umask'
```

```
mkdir path                         Create remote directory
progress                           Toggle display of progress meter
put [-Ppr] local [remote]          Upload file
pwd                                Display remote working directory
rename oldpath newpath             Rename remote file
rm path                            Delete remote file
rmdir path                         Remove remote directory
symlink oldpath newpath            Symlink remote file
version                            Show SFTP version
!command                           Execute 'command' in local shell
!                                  Escape to local shell

sftp> ls
Desktop    Documents  Downloads  Music      Pictures   Public     Templates
Videos     aliases    file.dat

sftp> mkdir temp

sftp> get file.dat
Fetching /home/lsga/file.dat to file.dat
/home/lsga/file.dat                          100%   32     0.0KB/s   00:00

sftp> cd temp

sftp> put file.dat
Uploading file.dat to /home/lsga/temp/file.dat
file.dat                                     100%   32     0.0KB/s   00:00

sftp> exit
```

3.1 End the screen Session

```
$ exit
[screen is terminating]
~ssh$ exit
Connection to 192.168.1.44 closed.
$ exit
Connection to 192.168.1.100 closed.
$
login:
```

NOTE: 8-port ethernet switches do not work properly when the output of one port is fed directly back into the input of another port . . .