## Lab 4 : Daemons and processes

### Exercise 1. Exploring running processes

**1.1. Find out the total number of processes that are currently running and identify the 10 most CPU-intensive processes.**

```
Slackware [Running]
root      1499  0.0  0.4   4208  1016 ?        Ss   12:05   0:00 /usr/sbin/sshd
root      1506  0.0  0.2   1860   608 ?        Ss   12:05   0:00 /usr/sbin/acpid
81        1517  0.0  0.3   2652   808 ?        Ss   12:05   0:00 /usr/bin/dbus-d
root      1520  0.0  1.0  25560  2596 ?        Ssl  12:05   0:00 /usr/sbin/conso
82        1589  0.0  2.0  15596  5020 ?        Ssl  12:05   0:00 /usr/sbin/hald
root      1644  0.0  1.0  21524  2644 ?        Sl   12:05   0:00 /usr/libexec/po
root      1662  0.0  0.4   3672  1200 ?        S    12:05   0:00 hald-runner
root      1694  0.0  0.4   3768  1028 ?        S    12:05   0:00 hald-addon-inpu
root      1706  0.0  0.4   3772  1024 ?        S    12:05   0:00 hald-addon-stor
82        1708  0.0  0.4   3576  1144 ?        S    12:05   0:00 hald-addon-acpi
root      1727  0.0  0.2   2080   660 ?        Ss   12:05   0:00 /usr/sbin/crond
daemon    1729  0.0  0.1   2072   320 ?        Ss   12:05   0:00 /usr/sbin/atd -
root      1741  0.0  0.4   2768  1064 ?        S<   12:05   0:00 /sbin/udevd --d
root      1754  0.0  0.1   2068   376 ?        Ss   12:05   0:00 /usr/sbin/gpm -
bob       1756  0.0  0.7   3360  1820 tty1     Ss   12:05   0:00 -bash
root      1757  0.0  0.2   1852   520 tty2     Ss+  12:05   0:00 /sbin/agetty 38
root      1758  0.0  0.2   1852   524 tty3     Ss+  12:05   0:00 /sbin/agetty 38
root      1759  0.0  0.2   1852   520 tty4     Ss+  12:05   0:00 /sbin/agetty 38
root      1760  0.0  0.2   1852   520 tty5     Ss+  12:05   0:00 /sbin/agetty 38
root      1761  0.0  0.2   1852   524 tty6     Ss+  12:05   0:00 /sbin/agetty 38
root      1788  0.0  0.0      0     0 ?        S    12:25   0:00 [flush-8:0]
bob       1803  0.0  0.3   2732   960 tty1     R+   12:28   0:00 ps aux
bob@Bry021:/home/bob$ ps aux | wc -l
91
bob@Bry021:/home/bob$ _
Left ⌘
```

```
Slackware [Running]
Tasks:  92 total,   1 running,  91 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.0%us,  0.3%sy,  0.0%ni, 99.7%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:    243708k total,   165420k used,    78288k free,    32728k buffers
Swap:  2815388k total,        0k used,  2815388k free,   105420k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 1780 bob       20   0  2628 1056  820 R  0.7  0.4   0:00.02 top
    1 root      20   0   824  276  236 S  0.0  0.1   0:00.89 init
    2 root      20   0     0    0    0 S  0.0  0.0   0:00.00 kthreadd
    3 root      20   0     0    0    0 S  0.0  0.0   0:00.00 ksoftirqd/0
    4 root      20   0     0    0    0 S  0.0  0.0   0:00.00 kworker/0:0
    5 root      20   0     0    0    0 S  0.0  0.0   0:00.00 kworker/u:0
    6 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 migration/0
    7 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 cpuset
    8 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 khelper
    9 root      20   0     0    0    0 S  0.0  0.0   0:00.00 kworker/u:1
   12 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 netns
  274 root      20   0     0    0    0 S  0.0  0.0   0:00.00 sync_supers
  276 root      20   0     0    0    0 S  0.0  0.0   0:00.00 bdi-default
  278 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 kblockd
  280 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 kacpid
  281 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 kacpi_notify
  282 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 kacpi_hotplug
  324 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 ata_sff
bob@Bry021:~$ _
Left ⌘
```

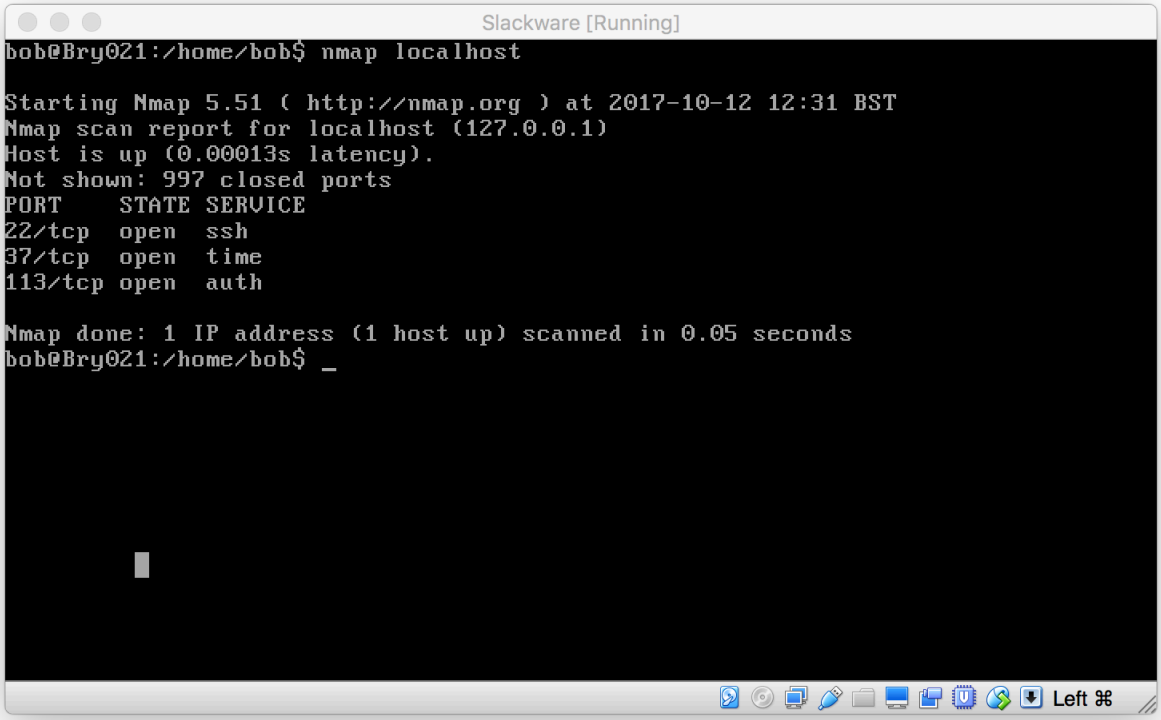
The number is provided by the command ps aux | wc -l

See last page for a description for those 10 processes.

Exercise 2. Exploring network processes

**2.1. Execute the command nmap localhost. Write down the processes returned.**

The command **nmap** (network mapper) is an utility that uses raw IP packets to determine what hosts are available on the network, what services those hosts are offering, what OS they are running, etc. It was designed to rapidly scan large networks, but works fine against single hosts. Many sysadmins also find it useful for tasks such as network inventory, managing service upgrade, etc. For example to see the services our machine is offering (three processes):



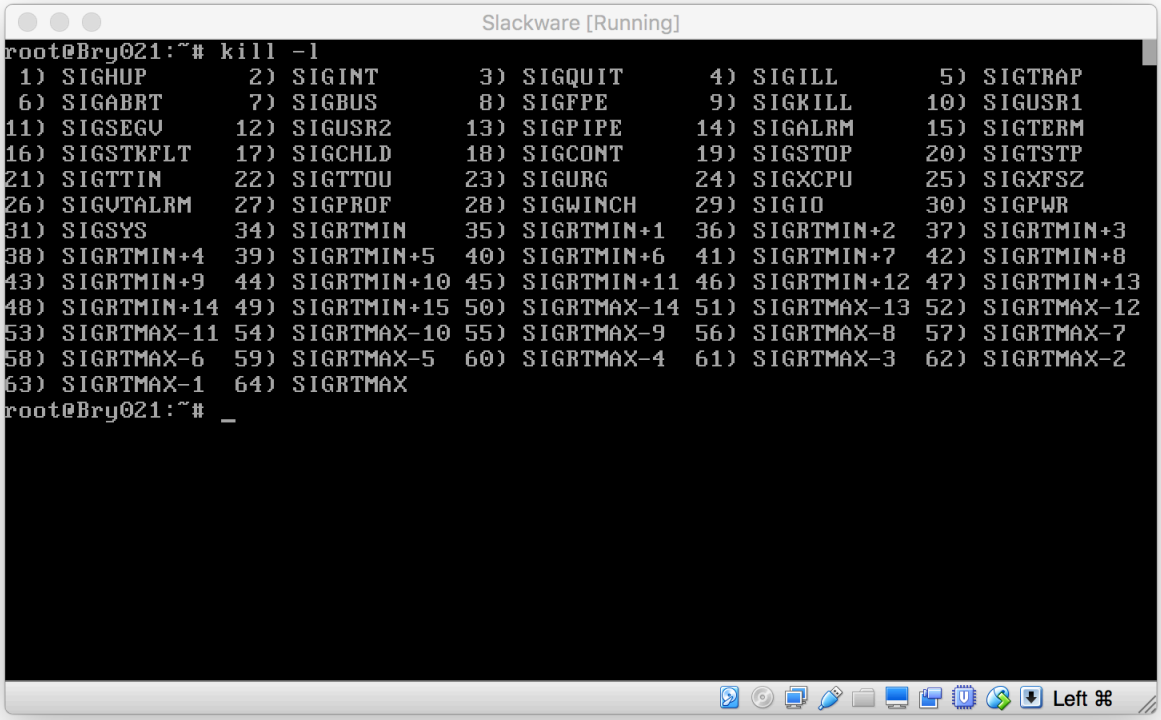

**37/tcp time** - server included in a set called "tcp small services". It's deprecated so it is going to be commented in /etc/inetd.conf (exercise 4).

**113/tcp auth** - server to identify the user behind a particular TCP connection.

**22/tcp ssh** - server to enable secured remote access (sshd is the server daemon)

Exercise 3. Exploring UNIX signals

The kill command sends signals to running processes in order to request their termination:




| Signal | No | Descritpion |
|---|---|---|
| SIGHUP | 1 | Hangup detected on controlling terminal (restart) |
| SIGINT | 2 | Interrupt from the keyboard |
| SIGQUIT | 3 | Quit from keyboard |
| SIGILL | 4 | Illegal instruction |
| SIGTRAP | 5 | Trap |
| SIGABRT | 6 | Abort signal |
| SIGBUS | 7 | Bus |
| SIGFP | 8 | Float point exception |
| SIGKILL | 9 | The process must quit immediately and will not perform any clean-up operations. |
| SIGUSR1 | 10 | User defined signal |
| SIGALRM | 14 | Alarm clock (timers) |
| SIGTERM | 15 | Software termination signal (sent by default) |
| SIGSTOP | 17,19,23 | Stop process |
| SIGCONT | 19,18,25 | Continue if stopped |

Exercise 4. Processes and networking

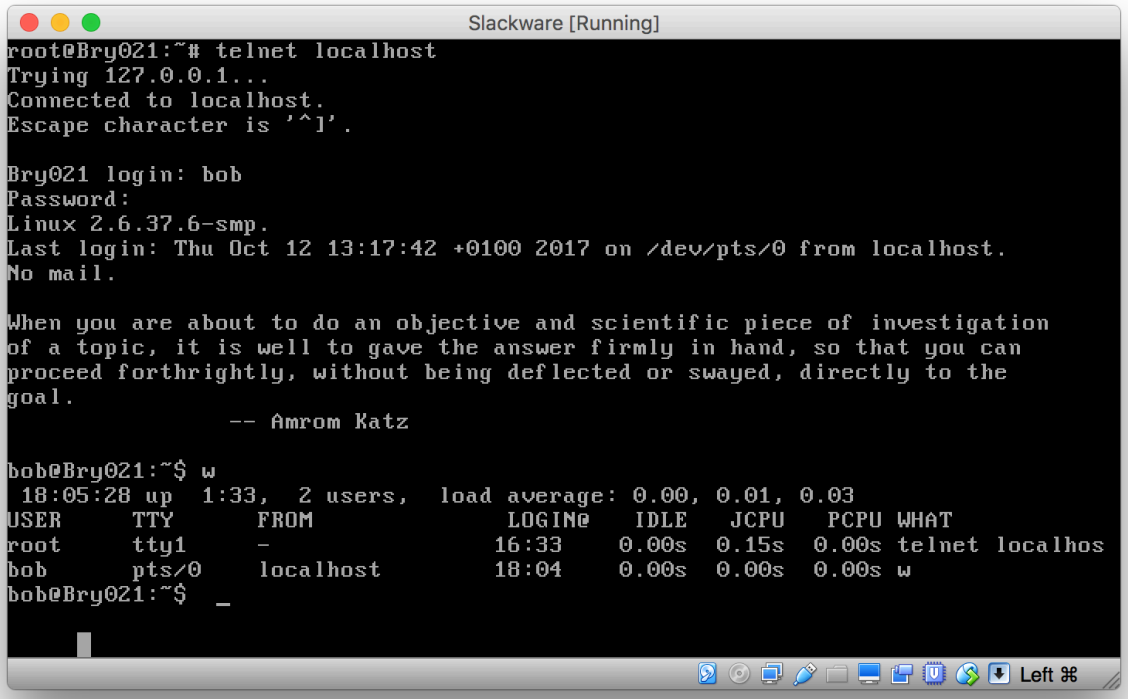
### 4.1. Edit /etc/inetd.conf to enable telnet and ftp

The daemon **inetd** provides Internet services. For each configured service, it listens for requests from connecting clients. Requests are served by spawning a process which runs the appropriate executable, but simple services such as *echo* are served by inetd itself.

Let's first remove the service in port 37. We are going to edit the file **/etc/inetd.conf.** Once it has been updated, the **inetd** service is restarted running this:

#./**etc/rc.d/rc.inetd** restart

Note that all processes listed in the /etc/rc.d/ directory start with the prefix "**rc"** (rc.inetd).

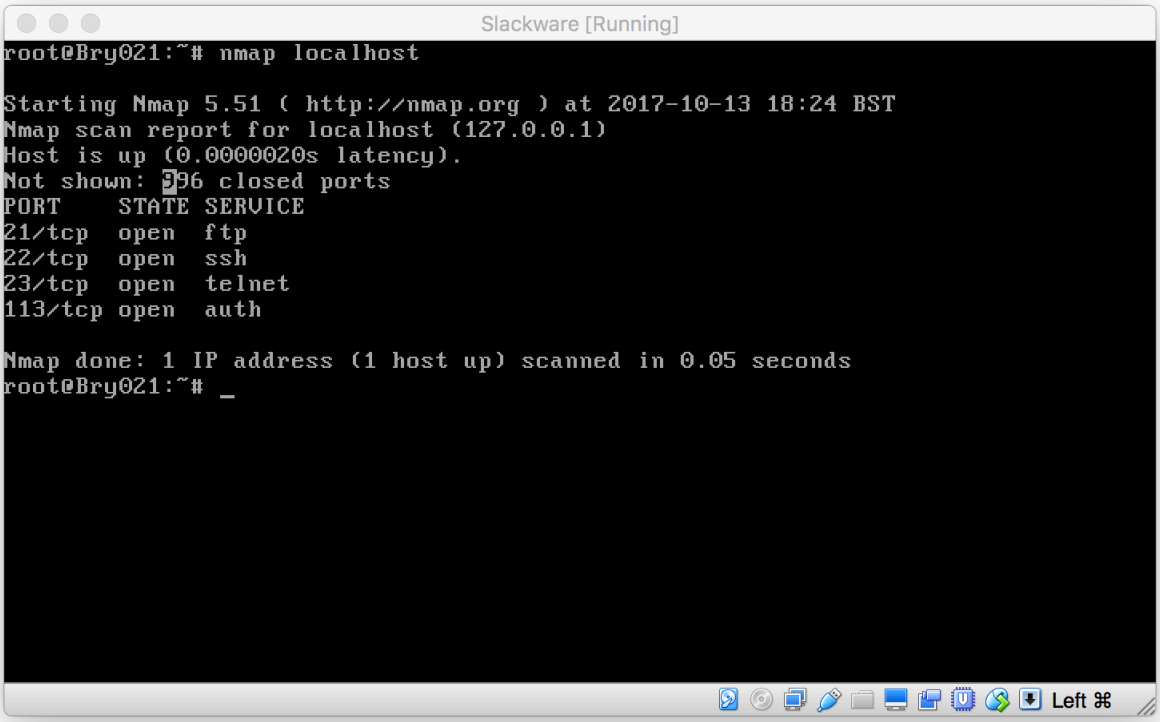


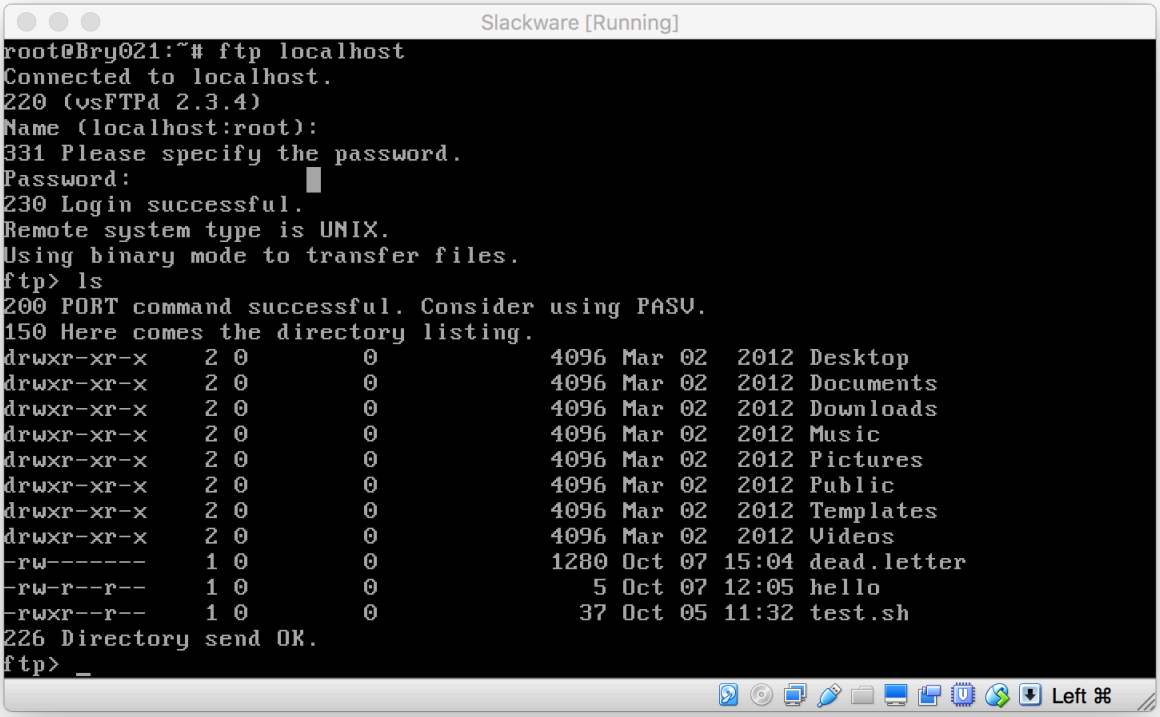

Therefore bob can login by using telnet. The command "**w**" shows that there are two users logged in the machine: root at tty1 (running telnet) and bob at pts/0 (running w).

The main reason of using the SSH protocol instead is because telnet sends data in plain text. SSH uses a public key for authentication.

Now let's enable ftp (port 21). We are going to use the **vsftpd** daemon. Its configuration files is located at /**etc/vsftp.conf.** There are two changes:

- Change **anonymous_enable=YES** to **NO** to disable Anonymous FTP.
- Uncomment **local_enable=YES** and **write_enable=YES** to allow you to make changes to the FTP server.



| Command | FTP operation |
|---|---|
| **put** | copy one file from the local machine to the remote machine |
| **get** | copy one file from the remote machine to the local machine |
| **ls, mkdir, rmdir, cd** | remote file-folder management |
| **lcd** | same as cd in local |

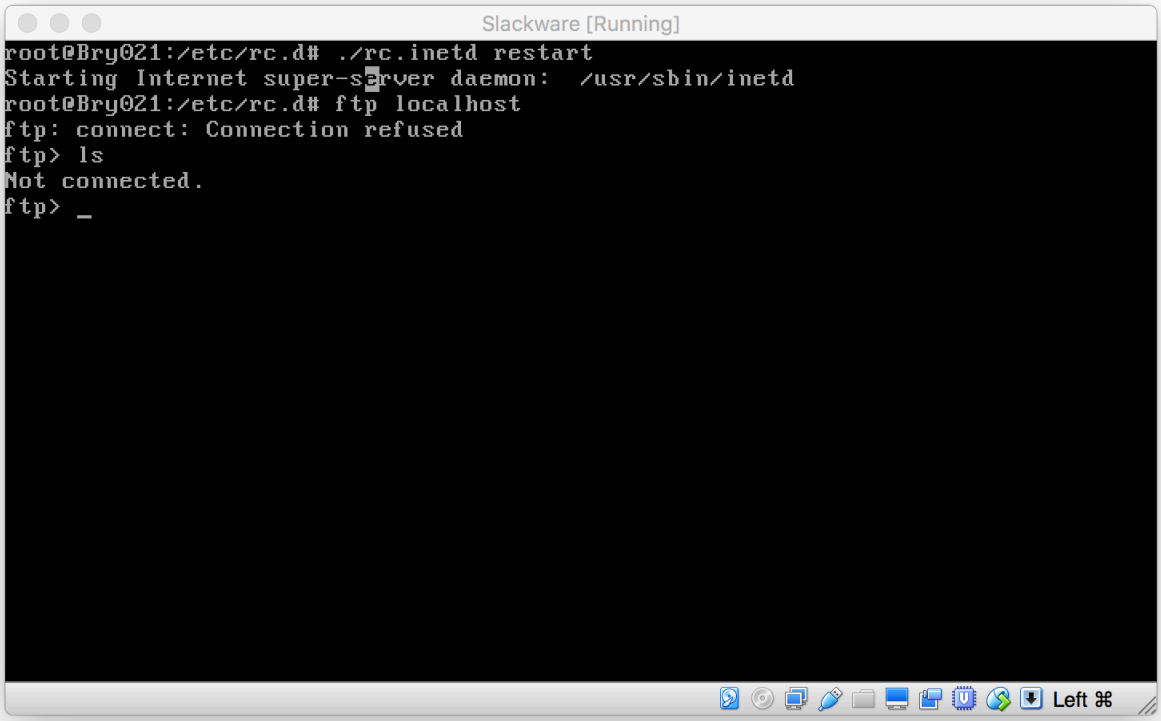
More FTP commands can be found at https://www.cs.colostate.edu/helpdocs/ftp.html.

### 4.2. Edit /etc/inetd.conf again to disable ftp.

Once the line is commented in the config file, the process should be restarted:

**# ./etc/rc.d/rc.inetd** restart

Now the connection is refused (ftp localhost).



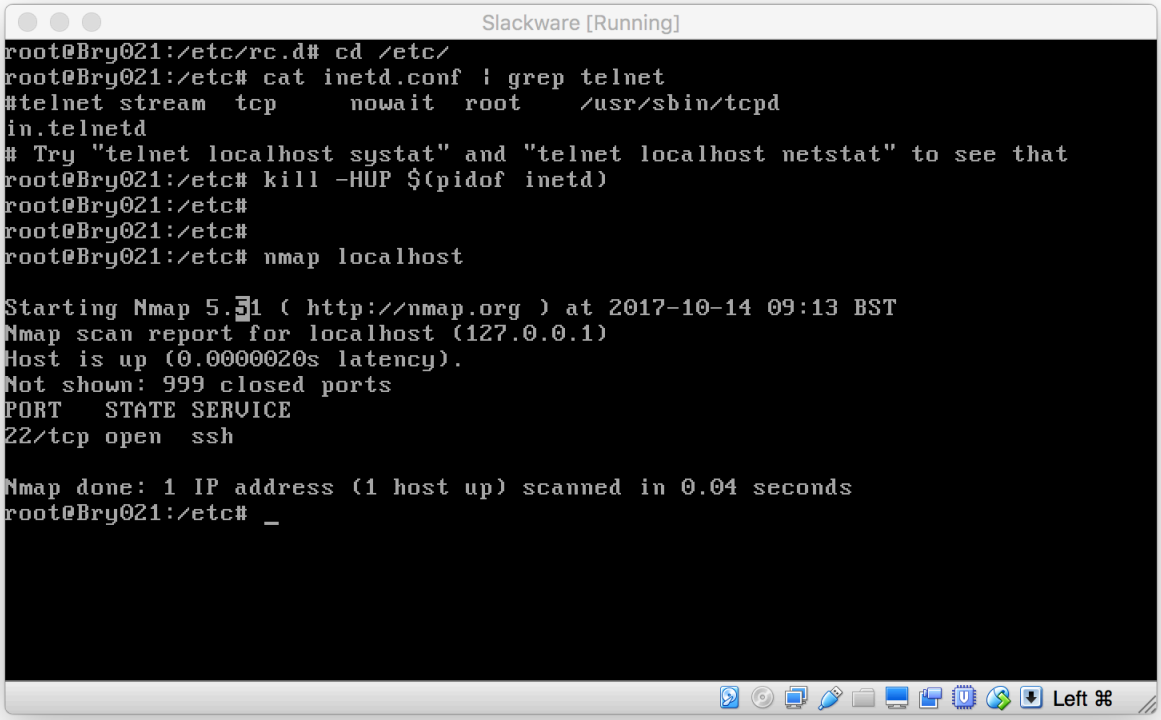
Exercise 5 (optional).

**5.1. How could you combine pidof with kill -HUP in order to restart inetd in one command?**

We can combine those commands in one line using a mini bash script (variable):

**# kill -HUP $(pidof inetd)**

The command in between brackets is evaluated first.




**5.2. Can you ftp as root? Is a good idea?**

Looking at the exercise 3, the root user is able to login in the FTP server but it's not a good idea. In this forum there are some clues:

https://askubuntu.com/questions/16178/why-is-it-bad-to-log-in-as-root

Basically it's like arming a little kid with an AK47, while he can happily play with his paintball gun.

### 5.3. From you host machine, telnet to your Slackware VM?

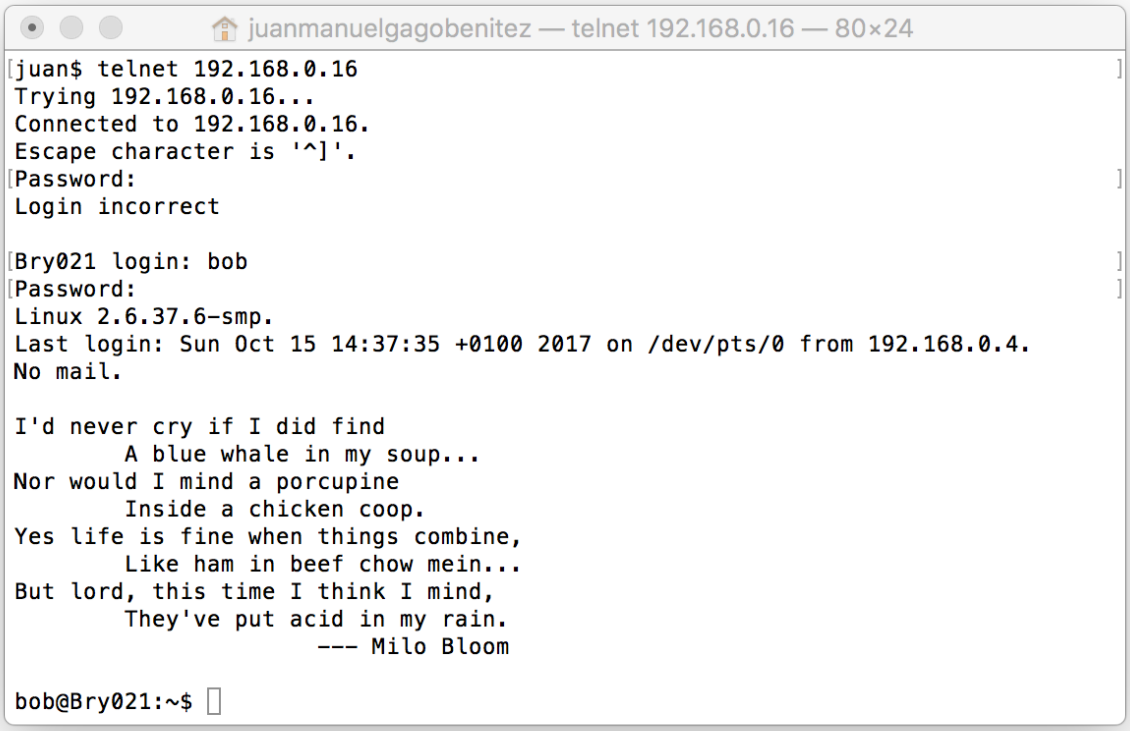
There are several options in the VirtualBox for networking (table 6.1)

| | VM — Host | VM1 — VM2 | VM —> Internet | VM <— Internet |
|---|---|---|---|---|
| Host Only | Yes | Yes | - | - |
| Bridged(*) | Yes | Yes | Yes | Port forwarding |
| NAT | - | - | Yes | Port forwarding |
| NAT Network | - | Yes | Yes | Port forwarding |

The default VirtualBox adapter is a NAT type (Network Address Translation) but a Bridged one is more convenient as our DHCP server (real router) will assign a real IP address to the VM.

Now let's make telnet and ftp servers accessible from any device in the network with port forwarding (still networking tab on VirtualBox):

| Service | Host IP | Host Port | Guest IP | Guest Port |
|---|---|---|---|---|
| Telnet | - | 23 | - | 23 |
| FTP | - | 21 | - | 21 |
| Apache | - | 80 | - | 80 |




More info at https://www.howtogeek.com/122641/how-to-forward-ports-to-a-virtual-machine-and-use-it-as-a-server/

**Answer 1.1:**

1. **init** (short for *initialization*) is the first process started during booting of the computer system. Init is a daemon process that continues running until the system is shut down.

2. **kthreadd** is a thread that kernel uses to spawn newer threads if required.

3. **ksoftirqd** is a per-cpu kernel thread that runs when the machine is under heavy soft-interrupt load.

4. **kworker/0:0** is placeholder for kernel worker threads, which perform most of the processing for the kernel, especially in cases where there are interrupts, timers, IO, etc.

5. **migration/0** distributes workload across CPU cores. You should have one migration process per processor core.

6. **cpuset** confines process to the processor and memory node subset. It is mounted in /dev/cpuset

7. **khelper :** module used to make calls to userspace implementations of what would be kernel modules.

8. **netns :** network namespace management is logically another copy of the network stack, with its own routes, firewall rules, and network devices.

9. **sync_supers** : Apache service (user "nobody")

10. **bdi-default:** breakpoint debugger interface.