How are the labs going?
How are the labs going?

Everybody can access workstations in Class 123?
How are the labs going?

Everybody can access workstations in Class 123?

Everybody obtained personal certificate for VPN?
How are the labs going?

Everybody can access workstations in Class 123?

Everybody obtained personal certificate for VPN?

Everybody configured VPN client at home?
How are the labs going?

Everybody can access workstations in Class 123?
Everybody obtained personal certificate for VPN?
Everybody configured VPN client at home?
Everybody can access personal profile over FTP?
How are the labs going?

Everybody can access workstations in Class 123?
Everybody obtained personal certificate for VPN?
Everybody configured VPN client at home?
Everybody can access personal profile over FTP?
Everybody can run lab virtual instance at home?
How are the labs going?

so far we received 96/99 certificate requests.

~$ echo $(($(cat /var/mail/ca | grep From: | wc -l)-1))
96

19/99 students accessed workstations only once

~# ls -alp /storage/home | grep 'Feb 1[123]' | wc -l
19

21/99 students accessed Lab Network from home

~# cat /var/log/openvpn.log | grep 2013 | \ 
  grep 'Feb [12][8901]' | \ 
  grep 'VERIFY OK: depth=0' | \ 
  cut -d' ' -f7 | grep '_' | cut -d'/\' -f1 | \ 
  sort | uniq | wc -l
21
How are the labs going?

Everybody can access workstations in Class 123?

Everybody obtained personal certificate for VPN?

Everybody configured VPN client at home?

Everybody can access personal profile over FTP?

Everybody can run lab virtual instance at home?

Constant activity in Lab Network (VPN), FTP is being most used protocol so far in the lab.
How are the labs going?

Everybody can access workstations in Class 123.

Everybody obtained personal certificate for VPN.

Everybody configured VPN client at home.

Everybody can access personal profile over FTP.

Everybody can run lab virtual instance at home.

Constant discussion on our QT board:

You need to admit that Linux users are used to solve such problems in comparison to Mac and Windows users ;)
If your quest will lead you to the dead end, try to contact Artjom Lind via his email. He can provide you with some important tips.

Well, with the same logics Windows users don't need manual for Windows too ;)
OK, challenge accepted.

Unfortunately not. As you use Linux daily as your main OS I guess it would not be a problem for you. Just check existing manuals for required files.

Is there any guide at the course's page on how to establish required VPN connection under Linux?

Yes, there is a lab session today (Feb 13 2013). Course page is maintained daily to reflect the latest state and you can trust info found here:
https://courses.cs.ut.ee/2013/syshald
And once again ...

Course Web Page:
https://courses.cs.ut.ee/2013/syshald

QT Board:
http://www.quicktopic.com/50/H/WTy6tmkWYapgW/p19.3#QTmsg11

VPN Server:
cisco-nat.mt.ut.ee (193.40.37.136)

FTP Server (over VPN or from Class 123):
192.168.10.1
OpenVPN manual

Windows:
https://courses.cs.ut.ee/2013/syshald/Main/WinXPVpnConf

MacOSX:
https://courses.cs.ut.ee/2013/syshald/Main/MacVpnConf

GNU/Linux:
https://courses.cs.ut.ee/2013/syshald/Main/LinuxVpnConf

Special thanks to one of you who agreed to share own configuration and write this manual. +1
We appreciate

Independent work
• Ability to share the knowledge with others
• Patience
• Attention

Feedback from you

Communication is the key
Anyway ...

- **Independent work:** **92 hours**
  - **Reading** Lecture and Lab materials
  - **Reading** related books
  - **Reading** related articles and journals
  - **Reading** UNIX documentation
  - **Reading** your own configuration!
  - **Administrating** your server
  - **Tracing errors** in your configuration
Otherwise ...

- Independent work: **92 hours**

  Reading ...
  Administrating your server
  Administrating ...
  Administrating ...
  Tracing errors in your configuration
  Tracing errors ...
  Tracing errors ...
  Tracing ...
  Tracing ...
  ...

Artjom Lind (artjom.lind@ut.ee)
21.02.2013
https://courses.cs.ut.ee/2013/syshald
In UNIX every single character is important

~~$ cd /home/user vs. $ cd home/user ~~

~~$ mkdir /lab1 vs. $ mkdir lab1 ~~

~~$ ls -lab vs. $ ls lab ~~

~~$ cat f1.txt > f2.txt ~

~~$ cat f1.txt >> f2.txt ~

~~$ du -hs * vs. du -hs ~
Basic UNIX knowledge

- Multitasking, multiuser system
- Layered architecture
  - Kernel
  - Shell
  - GUI
- File based system configuration
- Lots of shell commands
Users, groups

- Each user has UID
- Each group has GID
- Root has UID=0
- System users have UID<1000
- Regular users have UID>=1000
- Each user has one primary group
- Each user may belong to multiple secondary groups
Users, groups

- *id* - check user UID,GID and his secondary groups

  ```
  ~$ id user
  uid=1000(user) gid=1000(user)
  groups=1000(user)
  
  ~$ id devel
  uid=6666(devel) gid=6666(devel)
  groups=6666(devel),4(adm),6(disk),20(dialout),21(fax),24(cdrom),26(tape),27(sudo),29(audio),30(dip),44(video),46(plugdev),108(netdev),111(lpadmin),115(powerdev),116(scanner),121(fuse),125(scard),142(libvirt),1000(user),1009(vbox users)
  ```
File system
Mount points

System Root

User Profiles

Boot loader files and Linux kernel

System Root

/ /bin /etc /usr /var

/home

/boot

/dev/sda

sda1 sda2 sda3 sda4

SWAP partition

Artjom Lind (artjom.lind@ut.ee)

21.02.2013

https://courses.cs.ut.ee/2013/syshald
Mount Points

• Configured in /etc/fstab

• ~$ cat /etc/fstab

• # /etc/fstab: static file system information.

• #

• # <file system> <mount point> <type> <options>       <dump> <pass>

• /dev/sda1    swap     swap sw         0  0
• /dev/sda2    /        ext4 errors=remount-ro 0  1
• /dev/sda3    /home    ext4 defaults     0  2
• /dev/sda4    /boot    ext2 defaults     0  0
File System

- Root directory “/"
  - cd /
- Current directory
  - pwd
- Home directory “~"
  - cd ~
  - cd
- Parent directory “.."
  - cd..
- Sub-directory /home and /home/user
File parameters

- Owner
- Group
- Permission bits
- Date Modified

- Name
- Type
  - File, Directory
  - Link, FIFO, Block device

- Size

```
~$ ls -l vpn_manual.txt
-rw------- 1 devel devel 4740 Feb 20 14:39 vpn_manual.txt
~$ ls -l
-drwxr-x--- 10 devel devel 4096 Feb 21 06:02 Downloads
~$ ls -l text.txt
lrwxrwxrwx 1 devel devel 19 Feb 21 06:02 text.txt -> vpn_manual.txt
```
File Name in UNIX

• Case sensitive
  - Downloads and downloads are two different files
• Hidden files start with “.” (use `ls -la`)
• File name max length 255 bytes
• File path max length none
• Reserved characters:
  & ; | * ? ` " [ ] ( ) $ < > { } % ! # @ \
• Escaping reserved chars: `\( \) \# \! \?`
Permissions

• Three types of permissions:
  - r – read: file or directory read access
  - w – write: changing file or directory:
    • add or remove files from directory
  - x – execute: running application file or command, in case of directory: enter or list directory contents

• The permissions can be changed by file owner or by superuser (root)
Permissions

- -r--------  400 u+r
- --w--------  200 u+w  Owner user (u)
- ---x------- 100 u+x
- ----r------  040 g+r
- -----w-----  020 g+w  Owner group (g)
- ------x-----  010 g+x
- -------r----  004 o+r
- --------w----  002 o+w  All others (o) - world
- --------x-----  001 o+x
Changing permissions

- 
  - chmod g-rwx,o-rwx f1.txt
  - chmod go-rwx f1.txt
  - chmod a-x f1.txt
  - chmod u+rw,g+rx f1.txt

- 
  - chmod 600 f1.txt
    400+200 user rw
  - chmod 644 f1.txt
    400+200+40+4
    user rw, group r, other r
  - chmod 640 f1.txt ??

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>u</td>
<td>user</td>
</tr>
<tr>
<td>g</td>
<td>group</td>
</tr>
<tr>
<td>o</td>
<td>other</td>
</tr>
<tr>
<td>a</td>
<td>all</td>
</tr>
<tr>
<td>r</td>
<td>read</td>
</tr>
<tr>
<td>w</td>
<td>write (and delete)</td>
</tr>
<tr>
<td>x</td>
<td>execute (and access directory)</td>
</tr>
<tr>
<td>+</td>
<td>add permission</td>
</tr>
<tr>
<td>-</td>
<td>take away permission</td>
</tr>
</tbody>
</table>
Command-line interpreters

- **Shell**

- Wiki: A command-line interface (CLI) is a means of **interaction with a computer program** where the user (or client) issues commands to the program in the form of successive lines of **text** (command lines).

- **sh, ksh, csh, zsh, tcsh, bash**

- **bash** is default shell in GNU/Linux
Command line

• General form:

~$ command -options parameters

• Options alternate or specify the command behavior

• Parameters provide the input data by values, or by giving the path to the file. Parameter values may also alternate command behavior
Command line

• ~$ `head vpn_manual.txt`

Manual for SA13 Configuration on GNU/Linux

Configuring the OpenVPN

* in this manual "username" refers to your username in this course (in class 123)

To configure your openvpn you need certificates generated in first practical session.

1. First you need to install openvpn package. Install version not newer than 2.2 as 2.3 has a bug.

For that either use command

# apt-get install openvpn

...

• ~$ `head -n 1 vpn_manual.txt`

Manual for SA13 Configuration on GNU/Linux
Command Line

• Help option “-h” or “--help”
  - Gives brief description of command usage

• ~$ tail --help

Usage: tail [OPTION]... [FILE]...
Print the last 10 lines of each FILE to standard output.
With more than one FILE, precede each with a header giving the file name.
With no FILE, or when FILE is -, read standard input.
Mandatory arguments to long options are mandatory for short options too.
  -c, --bytes=K           output the last K bytes; alternatively, use -c +K to output bytes starting with the Kth of each file ....
Command Line

• Read detailed manual for specific command using UNIX online documentation or so called manual (man) pages.

• man - special command to view manual of specific command:
  • ~$ man command
  • ~$ man tail
    – Gives detailed description of “tail” command
    – Navigate with arrow keys, terminate with “q” key
Command Line (\textit{shell})

- **Internal commands:**
  - \textit{shell} built-in commands
  - Examples:
    - \texttt{cd} \hspace{2mm} \texttt{echo} \hspace{2mm} \texttt{exit}

- **External commands:**
  - OS level programs, located in /bin and /usr/bin
  - For exact location of the external program use “which” command:
    - \texttt{~$ which ls}
      - /bin/ls
    - … returned the location of “ls”
Command Line (*shell*)

- Important meta symbols:
  - File name patterns:
    - `*` - any amount of any symbols
    - `?` - exactly one symbol
    - `[symbols]` – exactly one symbol from the list
  - `-Escaping`
    - `\( \)` `\$ ` `\#` ...
  - `command` or `$(command)` – in place execution
    - Expression will be replace with the output of the command
File management commands

- `pwd` – print current directory
- `cd dir` – change directory
- `cp -r dir1 dir2` – directory copy
- `cp file1 file2 ... fileN directory` – copy files to directory
- `mv source_file target_file` - move/rename file
- `mv file1 file2 ... fileN directory` – move files to directory
- `rm file1 file2` – delete files
- `rm -r directory` – delete directory
File permissions and ownership

- `chmod` – change permissions
  - Covered in slides 26-28

- `chown` – change ownership
  - `chown newuser.newgroup file1 file2 ... fileN`
    - Change ownership of the files to newuser and newgroup
  - `chown -R newuser.newgroup directory`
    - Change ownership of the directory and all the files within recursively
Process Management

- `ps` – view process table
- `ps -e` - view all (also detached and not owned)
- `ps -f` – detailed table
- `ps -u user` – view processes of user

- `top, htop` – interactive process tables
Process Management

- It is possible to send the signals to process by process id (PID)
  - `kill -SIG PID` (send signal)
  - `kill -1` (view signals)
  - `kill -15 PID` (SIGTERM) – soft terminate
  - `kill -9 PID` (SIGKILL) – hard terminate
- And by process name
  - `pkill -15 xclock` soft terminate xclock process
- In graphical environment
  - `xkill` – and the just select the window you want to terminate
Process Management

- Process priorities in UNIX
  - Highest -20
  - Lowest 19
  - Default 10
- Superuser permission required to set process priority below 0
- `nice -n N` – run process with N priority
- `renice -n N` – change priority of running process
Process Management

- **nohup** – run program detached (in background)
- **Ctrl+C** in runtime – send SIGINT (2) signal
- **Ctrl+Z** in runtime – suspend and detach process
- **bg** – send detached process to background
- **fg** – restore detached process from background
Disk management and quotas

- Disk Quota
  - Limits user disk usage
    - Number of files
    - Amount used disk space
  - Soft quota
  - Hard quota

- `quota`
- `quota -v` detailed quota info
- Class 123 quota is 50G per user
Disk management and quotas

- `du` – Disc Usage information (sum of all files in current directory)
- `du -sk * .??* | sort -n` – sorted list of disc usage by individual files and directories
- `df` – File system load information
Default Communication Channels

- STDIN (0) – Standard Input
- STDOUT (1) – Standard Output
- STDERR (2) – error reports
- Each program apart from options and parameters can read from STDIN and write to STDOUT, STDERR
- STDIN comes by default from command line input
  - Keyboard
- STDOUT, STDERR – by default command line output
  - Terminal application (command prompt)
Default Communication Channels

- ```~$ N=''; \n    echo 'Say something :'; \n    read N; \n    echo 'You said: ' $N;```  

  - Here we ask user to input some data, obviously `read` command will be expecting data from STDIN letting user type in some words.
  - `echo` prints the corresponding messages to the STDOUT, so user can read the from command prompt.
  - `N` variable is only user to store the user input.
  - Here we do not use STDERR
Input/Output Redirection

- `command1 > fail1`
  - Write output of the command1 to file1
- `command1 >> fail1`
  - Append output of the command1 to file1
- `command1 < fail1`
  - Command1 input from file1
- `2>&1`
  - Redirect STDERR as well as the STDOUT
Input/Output Redirection

- ~$ echo 'Input some words, Enter to separate.' \  
  echo 'Ctrl+C to finish : '; \  
  cat > words.txt;
- Here we ask user to provide list of words (in several lines), and we write the word list int file words.txt. File writing occurs by redirecting output of “cat” command.

- ~$ echo 'Sorted list :'; \  
  sort < words.txt
- Now we show the alphabetically sorted list of words by providing the input to sort command from the file we just have written.
Command pipelines

- Powerful utility to combine simple utility
- Pipeline redirects the standard output of one command to the standard input of the next command.
- Commands in pipeline run parallel in separate processes
- No need for temporary files
- `du -sk * .??* | sort -n`
  - Do you remember this expression?
Command pipelines

- `command1 | command2`
- the pipe “|” is one way
- synchronization
  - Command2 waits if pipe is empty
  - Command1 waits if pipe is full
  - Command2 reads if pipe not empty
  - Command1 writes if pipe not full
- possible errors
  - pipe lock
  - broken pipe
Some useful links