Operating Systems – Linux 1-2 Measurements – Background material

Introduction

The Linux measurements were designed to allow you to have an impression about the administration of Linux severs along with providing practical experiences on the most common administration tasks. This document summarizes all background knowledge required to accomplish the measurements. The possible test questions can also be found aligned to the right.

Installing the operating system

During the measurement the VMWare Player application will be used as virtualization platform. This application can be downloaded from the VMWare homepage for free[1]. When you click your mouse into the virtual machine running in the VMWare Player the keyboard and the mouse are locked to the application. If you want to release back inputs for the host OS you have to press the **Ctrl+Alt** keys.

How can you release the mouse and keyboard locked by the guest OS in the virtual machine?

During the first measurement the OS will be installed from an ISO disc image. Regarding the installation process some decisions have complex background that is not discussed in the measurement guide to document. The most interesting questions will be highlighted in the following two chapters.

Automatic updates

For desktop operating systems automatic updates have been around for a while: most frequent desktop operating systems ship with automatic updates turned on, and the system installs available updates on its own and asks for permission to reboot the computer if necessary. This is the default behavior if the user does not explicitly change the settings.

The Ubuntu installer woks differently: the installer suggests disabling the automatic updates by offering the off option by default. The decision about automatic updates for server systems has to find a balance between two threats: when automatic updates are enabled the system will install software without the consent of administrators, and for a high reliability already working server the changes in the software environment might cause reliability issues. The other threat is related to security: without installing the updates the system might quickly become a target to online attacks whenever a new vulnerability is found. So to sum up, installing automatic updates decreases reliability but increases security.

The OS installer assumes that the newly installed server will be under constant administrator supervision, thus the administrators can quickly decide about each available update. However in

[1] https://my.vmware.com/web/vmware/downloads

most cases this assumption is wrong as most servers have no administrator personnel. Therefore we will enable the automatic updates during the installation process.

Why does the installer suggests automatic updates to be turned off?

Installing the boot loader

In the laboratory when the installer asks where to put the GRUB boot loader, we will allow it to use the Master Boot Record of the disk. In this particular situation it is a good idea, as there are no other installed operating systems in the virtual machine. However when the system is installed to a computer already having other operating systems caution is needed: in most cases GRUB identifies other installed operating systems, and is also able to boot them, but allowing to overwrite the MBR should only be confirmed if all other operating systems were found by the installer.

Using the Ubutnu server

The console

During both measurements all commands will be issued from the terminal: initially the console of the operating system will be used, but after installing an SSH (Secure Shell) server the SSH client application will be the preferred interface. Although by now all desktop operating systems and mobile devices use graphical user interfaces, the console access has still remained popular among system administrators working with Unix systems. Using the terminal has the following main advantages compared to a graphical interface:

- The terminal requires less resources compared to a graphical environment, thus the terminal can remain responsive under high system loads.
- The console can be accessed through low bandwidth connections when a graphical interface would lag noticeably.
- The shell running in the terminal helps every day work with a lot of built in function: configuration files are easy to edit, logs can be monitored and filtered in real time, inputs and outputs of different applications can be interconnected thus complex tasks can be written in a few lines of commands and without installing any additional software.
- Every command issued on the console by hand can be also written into a shell script without change: shell scripts can be recalled later and as the shell supports several programming formula (if-else branches, loops, etc.) many complex application is also implemented in the form of a shell script.

Of course there are several tasks much more suitable for a graphical user interfaces: editing images, formatted text, video or sound is much easier using a graphical interface – luckily these tasks are not typical in a server environment.

What are the advantages of terminal access compared to a graphical user interface in server environment?

Tips when using the shell:

- If you press **TAB**, the shell tries to complete your input if there is a command or file beginning with the entered text. This way pressing **TAB** can speed up notably your command input!
- You can reload previous commands using the **up/down** arrow keys of your keyboard. Editing these recalled commands is also possible.
- Searching in the command history can be initiated by pressing **CTRL+R**.

Users and permissions

The installer of the operating system is responsible for creating the first user. However this user in Ubuntu does not have root (unlimited) permissions for administering the system. While other Unix operating systems might allow root login and direct command execution, Ubuntu only allows restricted users to access the system, and elevated rights must be asked for individual commands whenever needed. The background of this restriction is related to the fact that if commands are executed in the shell with root privileges small mistakes can cause big trouble. For example see the following two commands:

- **rm –rf /tmp/folder**
- **rm –rf / tmp/folder**

These commands only differ in an almost unnoticeable space character. However the firs command deleted a folder under the tmp temporary partition, the second one erases the whole root partition, thus practically all files are deleted from the disc. When the second command is executed by a standard user the system returns permission denied error, and no harm is done.

Why do all users have restricted rights in Ubuntu?

Furthermore as almost all Unix system has its root user called “root”, each SSH server receives countless attempts trying to log in with this root user using easy to guess passwords – these attacks are originated by automatic hacking programs. To make such attacks unfeasible most SSH servers disable root access by default even if the root user is allowed to log in on the local terminal.

Why is root login disabled on most SSH servers?

As described above all users logging in into Ubuntu are restricted users. Every time root access is needed the **sudo** command must be used to provide elevated rights to the command entered after **sudo** on the same line. **Sudo** asks for your password when first used to ensure that the terminal is not hijacked by anybody else. To be able to execute commands with **sudo**, you have to be member of the user group named sudo, thus using this command is only allowed for the users being members of this special group.

What command can be used to execute other commands with root rights?

File system permissions

The traditional file system permission implementation of Unix systems categorizes users into three groups for each file: the owner of the file (only a single user), the owner group of the file (a single group that can be different from the group of the owner of the file), others (everybody else). Each group can have three different permissions: read, write and execute (execute for files means running them as programs, for folders means listing the contents of the folder). File system permissions can be seen on the output of the **ls -l** command below:

```
[eredics@morgo log]$ ls -l
total 13588
drwx------. 2 root   root      4096 Nov  1 03:50 aide
-rw-------. 1 root   root      2564 Jul 10  2012 anaconda.ifcfg.log
-rw-------. 1 root   root     19705 Jul 10  2012 anaconda.log
-rw-------. 1 root   root     38359 Jul 10  2012 anaconda.program.log
-rw-------. 1 root   root     96749 Jul 10  2012 anaconda.storage.log
-rw-------. 1 root   root    131134 Jul 10  2012 anaconda.syslog
-rw-------. 1 root   root     26752 Jul 10  2012 anaconda.xlog
-rw-------. 1 root   root      4949 Jul 10  2012 anaconda.yum.log
drwxr-xr-x. 2 root   root     45056 Mar  8 00:00 atop
drwxr-x---. 2 root   root      4096 Mar  4 09:47 audit
drwxr-x---. 2 bacula bacula    4096 Feb 19  2015 bacula
-rw-r--r--. 1 root   root      1994 Feb  1 16:28 boot.log
-rw-------. 1 root   utmp    218496 Mar  8 00:49 btmp
-rw-------. 1 root   utmp     24696 Mar  1 03:09 btmp-20160301.gz
drwxr-xr-x. 2 clam   clam      4096 Mar  1 03:09 clamav
-rw-------. 1 root   root    227823 Mar  8 08:15 cron
```

In the above example the **ls -l** command was executed in the log directory. Most files are owned by the user root and the group root, but there are some other folders owned by bacula or clam as well. Most files allow only read write access to their owner user and no rights to anybody else (rw------- code), while the directory **atop** can be both read and listed by the owner group and the others group too. As for the **bacula** folder the owner user can read, write and execute, and the owner group can read and execute.

What three groups can be file permissions granted for?

When setting the permissions using **chmod** a 3 digit code is used in most cases, just like **chmod 644 boot.log**. These three digits represent the three access groups for each file, and each single digit of the code represents the three possible operations:

- the MSB represents reading (value of 4),
- the next bit stands for writing (value of 2),
- the LSB is for executing (value of 1).

To allow reading for a group the digit can be calculated as 4 (reading) + 0 (no writing) + 0 (no executing). If reading and writing should be allowed the code is 4+2+0=6, and if all right should be allowed the code will be 7. Based on the above it is easy to see that the example command of **chmod 644 boot.log** sets the rights just as they were seen on the picture above.

How can you calculate the 3 digit number for setting file permissions with chmod?

Using the package manager

In today's Unix based system most software installation is performed by a package manager. Of course software can still be downloaded as source code and compiled in place, but in most cases already available compiled software packages provide convenient alternatives. Debian based systems

just like Ubuntu use the APT (Advance Package Tool) package manager. The 3 most used commands related to APT are the following

- **apt-get update**: Updates the local package catalog from the remote servers. This command does not install any software; the only goal is to synchronize the local package catalog with the online ones to make the system aware about new software packages or available updates.
- **apt-get upgrade**: This command actually downloads and installs software updates for all packages. In most cases it is run after the apt-get update command to keep the system as up-to-date as possible.
- **apt-get install**: This command is used to install new software packages to the system. Software dependencies are handled by the installer, thus all required packages are installed along with the selected ones.

As software installation and updates are administration tasks they require root rights. Therefore the **apt-get** commands are always used with **sudo** to avoid permission problems.

What is the command to install new software packages?

What is the difference between apt-get update and apt-get upgrade?

Basic Unix commands and directories

The commands and folders below are used in the laboratory, thus their short description can be a question in the beginning test.

Important commands

- **man**: displays the manual page for the command entered as argument
- **cd**: change directory
- **ls**: list directory contents
- **cp**: copy files or directories
- **mv**: move files or directories
- **rm**: remove files or directories
- **less**: display the contents of a text file
- **nano**: edit the contents of a text file
- **sudo**: run the following command on the same line with root permissions

What is the *** command used for?

Important directories

- **/home**: personal folders (containing personal file, the mailbox file, settings, etc.) for each user are located in this directory.
- **/etc**: all configuration files can be found under this directory
- **/var/log**: system logs and all other logs are kept in this directory

What is the *** directory used for?