“Київський фаховий коледж зв’язку”

Циклова комісія комп’ютерної та програмної інженерії

**ЗВІТ ПО ВИКОНАННЮ**

**ЛАБОРАТОРНОЇ РОБОТИ №8**

**з дисципліни: «Операційні системи»**

**Тема: «Збереження службових даних системи та її мережева конфігурація»**

Виконав студент

групи РПЗ-23А

Туровський В.

Перевірила викладач

Сушанова В.С.

Київ 2025

**Тема: “Збереження службових даних системи та її мережева конфігурація”**

**Мета роботи:**

1. Отримання практичних навиків роботи з командною оболонкою Bash.
2. Знайомство з базовими структурами для збереження системних даних - процеси, память, лог-файли та повідомлення про стан ядра.
3. Знайомство зі стандартом FHS.
4. Знайомство з діями при налаштуванні мережі.

**Матеріальне забезпечення занять:**

1. ЕОМ типу IBM PC.

2. ОС сімейства Windows та віртуальна машина Virtual Box (Oracle).

3. ОС GNU/Linux (будь-який дистрибутив).

4. Сайт мережевої академії Cisco netacad.com та його онлайн курси по Linux

**Короткі теоретичні відомості:**

**Where Data is Stored**

An implementation of the Linux kernel includes many subsystems that are a part of the kernel itself and others that may be loaded in a modular fashion when needed. Key functions of the Linux kernel include a system call interface, process management, memory management, virtual filesystems, networking, and device drivers.

In short, via a shell, the kernel accepts commands from the user and manages the processes that carry out those commands by giving them access to devices such as memory, disks, network interfaces, keyboards, mice, monitors and more.

A typical Linux system has thousands of files. The Filesystem Hierarchy Standard provides a guideline for distributions on how to organize these files. It is important to understand the role of the Linux kernel and how it both processes and provides information about the system under the /proc and /sys pseudo filesystems.

**Processes**

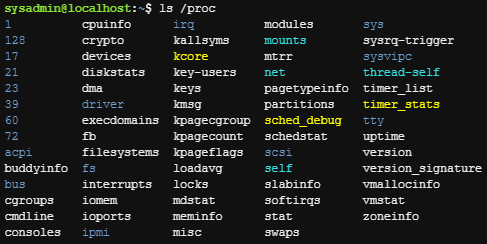
The kernel provides access to information about active processes through a pseudo filesystem that is visible under the /proc directory. Hardware devices are made available through special files under the /dev directory, while information about those devices can be found in another pseudo filesystem under the /sys directory.

Pseudo filesystems appear to be real files on disk but exist only in memory. Most pseudo file systems such as /proc are designed to appear to be a hierarchical tree off the root of the system of directories, files and subdirectories, but in reality only exist in the system's memory, and only appear to be resident on the storage device that the root file system is on.

The /proc directory not only contains information about running processes, as its name would suggest, but it also contains information about the system hardware and the current kernel configuration.

The /proc directory is read, and its information utilized by many different commands on the system, including but not limited to top, free, mount, umount and many many others. It is rarely necessary for a user to mine the /proc directory directly—it’s easier to use the commands that utilize its information.

See an example output below:



The output shows a variety of named and numbered directories. There is a numbered directory for each running process on the system, where the name of the directory matches the process ID (PID) for the running process.

For example, the numerals 72 denote PID 72, a running program, which is represented by a directory of the same name, containing many files and subdirectories that describe that running process, it’s configuration, use of memory, and many other items.

On a running Linux system, there is always a process ID or PID 1.

There are also a number of regular files in the /proc directory that provide information about the running kernel:

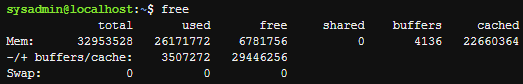
* /proc/cmdline - Information that was passed to the kernel when it was first started, such as command line parameters and special instructions;
* /proc/meminfo - Information about the use of memory by the kernel;
* /proc/modules - A list of modules currently loaded into the kernel to add extra functionality.

**Memory**

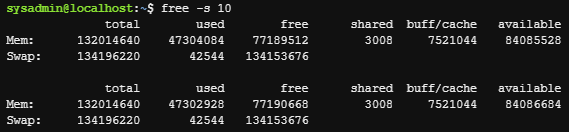
Memory on a modern Linux system is governed and managed by the kernel. The hardware memory on the system is shared by all the processes on the system, through a method called virtual addressing. The physical memory can be referenced by a number of processes, any of which may think they are able to address more memory than they actually can. Virtual addressing allows many processes to access the same memory without conflicts or crashes. It does this by allocating certain areas of a physical (or virtual) hard disk to be used in place of physical RAM. Memory is divided into blocks of equally sized units that can be addressed like any other resource on the system. Not only can the system access memory from local system addresses, but it can also access memory that is located elsewhere, such as on a different computer, a virtual device, or even on a volume that is physically located on another continent!

While a detailed review of Linux memory addressing is beyond the scope of this course, it’s important to note the difference between user space and kernel space. Kernel space is where code for the kernel is stored and executed. This is generally in a “protected” range of memory addresses and remains isolated from other processes with lower privileges. User space, on the other hand, is available to users and programs. They communicate with the Kernel through “system call” APIs that act as intermediaries between regular programs and the Kernel. This system of separating potentially unstable or malicious programs from the critical work of the Kernel is what gives Linux systems the stability and resilience that application developers rely on.

Executing the free command without any options provides a snapshot of the memory being used at that moment.



If you want to monitor memory usage over time with the free command, then you can execute it with the -s option (how often to update) and specify that number of seconds. For example, executing the following free command would update the output every ten seconds:



**Log Files**

As the kernel and various processes run on the system, they produce output that describes how they are running. Some of this output is displayed as standard output and error in the terminal window where the process was executed, though some of this data is not sent to the screen. Instead, it is written to various files. This information is called log data or log messages.

Log files are useful for many reasons; they help troubleshoot problems and determine whether or not unauthorized access has been attempted.

Some processes can log their own data to these files, other processes rely on a separate process (a daemon) to handle these log data files.

Logging daemons differ in two main ways in recent distributions. The older method of doing system logging is two daemons (named syslogd and klogd) working together, but in more recent distributions, a single service named rsyslogd combines these two functions and more into a single daemon.

In yet more recent distributions, those based on systemd, the logging daemon is named journald, and the logs are designed to allow for mainly text output, but also binary. The standard method for viewing journald-based logs is to use the journalctl command.

Regardless of what the daemon process being used, the log files themselves are almost always placed into the /var/log directory structure. Although some of the file names may vary, here are some of the more common files to be found in this directory:

| File | Contents |
| --- | --- |
| boot.log | Messages generated as services are started during the startup of the system. |
| cron | Messages generated by the crond daemon for jobs to be executed on a recurring basis. |
| dmesg | Messages generated by the kernel during system boot up. |
| maillog | Messages produced by the mail daemon for e-mail messages sent or received. |
| messages | Messages from the kernel and other processes that don't belong elsewhere. Sometimes named syslog instead of messages after the daemon that writes this file. |
| secure | Messages from processes that required authorization or authentication (such as the login process). |
| journal | Messages from the default configuration of the systemd-journald.service; can be configured in the /etc/journald.conf file amongst other places. |
| Xorg.0.log | Messages from the X Windows (GUI) server. |

You can view the contents of various log files using two different methods. First, as with most other files, you can use the cat command, or the less command to allow for searching, scrolling and other options.

The second method is to use the journalctl command on systemd-based systems, mainly because the /var/log/journal file now often contains binary information and using the cat or less commands may produce confusing screen behavior from control codes and binary items in the log files.

Log files are rotated, meaning older log files are renamed and replaced with newer log files. The file names that appear in the table above may have a numeric or date suffix added to them: for example, secure.0 or secure-20181103.

Rotating a log file typically occurs on a regularly-scheduled basis: for example, once a week. When a log file is rotated, the system stops writing to the log file and adds a suffix to it. Then a new file with the original name is created, and the logging process continues using this new file.

With most modern daemons, a date suffix is used. So, at the end of the week ending November 3, 2018, the logging daemon might stop writing to /var/log/messages (or /var/log/journal), rename that file /var/log/messages-20181103, and then begin writing to a new /var/log/messages file.

Although most log files contain text as their contents, which can be viewed safely with many tools, other files such as the /var/log/btmp and /var/log/wtmp files contain binary. By using the file command, users can check the file content type before they view it to make sure that it is safe to view. The following file command classifies /var/log/wtmp as data, which usually means the file is binary:



For the files that contain binary data, there are commands available that will read the files, interpret their contents and then output text. For example, the lastb and last commands can be used to view the /var/log/btmp and /var/log/wtmp files respectively.

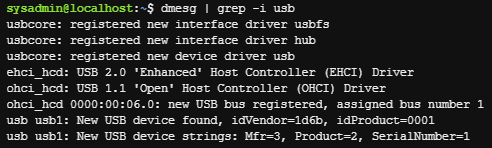
**Kernel Messages**

The /var/log/dmesg file contains the kernel messages that were produced during system startup. The /var/log/messages file contains kernel messages that are produced as the system is running, but those messages are mixed in with other messages from daemons or processes.

Although the kernel doesn't have its own log file normally, one can be configured for it by modifying either the /etc/syslog.conf file or the /etc/rsyslog.conf file. In addition, the dmesg command can be used to view the kernel ring buffer, which holds a large number of messages that are generated by the kernel.‌⁠​​⁠​

On an active system, or one experiencing many kernel errors, the capacity of this buffer may be exceeded, and some messages might be lost. The size of this buffer is set at the time the kernel is compiled, so it is not trivial to change.

Executing the dmesg command can produce up to 512 kilobytes of text, so filtering the command with a pipe to another command like less or grep is recommended. For example, if a user were troubleshooting problems with a USB device, then searching for the text USB with the grep command is helpful. The -i option is used to ignore case:



**Filesystem Hierarchy Standard**

Among the standards supported by the Linux Foundation is the Filesystem Hierarchy Standard (FHS), which is hosted at the URL http://www.pathname.com/fhs/.

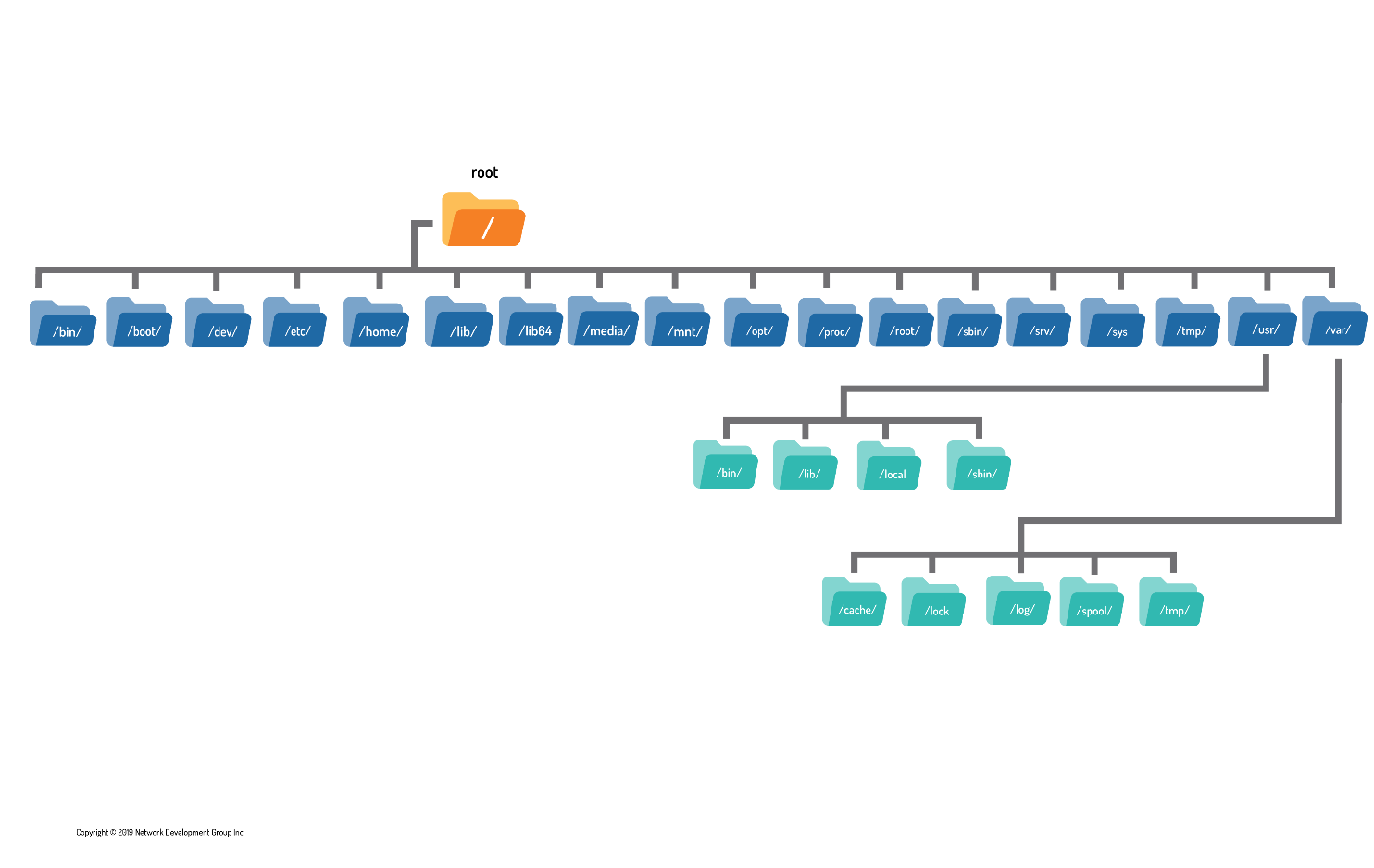
A standard is a set of rules or guidelines that it is recommended to follow. However, these guidelines certainly can be broken, either by entire distributions or by administrators on individual machines.

The FHS standard categorizes each system directory in a couple of ways:

A directory can be categorized as either shareable or not, referring to whether the directory can be shared on a network and used by multiple machines.

The directory is put into a category of having either static files (file contents won't change) or variable files (file contents can change).

To make these classifications, it is often necessary to refer to subdirectories below the top level of directories. For example, the /var directory itself cannot be categorized as either shareable or not shareable, but one of its subdirectories, the /var/mail directory, is shareable. Conversely, the /var/lock directory should not be shareable.



**Basic Network Terminology**

Before setting up a network or accessing an existing network, it is beneficial to know some key terms that are related to networking. This section explores the terms with which you should be familiar. Some of the terms are basic, and you may already be familiar with them. However, others are more advanced.

**Host**. A host is a computer. Many people automatically think of a desktop computer or laptop when they hear the term computer. In reality, many other devices, such as cell phones, digital music players and many modern televisions, are also computers. In networking terms, a host is any device that communicates via a network with another device.

**Network**. A network is a collection of two or more hosts (computers) that are able to communicate with each other. This communication can be via a wired connection or wireless.

**Internet**. The Internet is an example of a network. It consists of a publicly accessible network that connects millions of hosts throughout the world. Many people use the Internet to surf web pages and exchange emails, but the Internet has many additional capabilities besides these activities.

**Wi-Fi.** The term Wi-Fi refers to wireless networks.

**Server**. A host that provides a service to another host or client is called a server. For example, a web server stores, processes and delivers web pages. An email server receives incoming mail and delivers outgoing mail.

**Service**. A feature provided by a host is a service. An example of a service would be when a host provides web pages to another host.

**Client**. A client is a host that is accessing a server. When you are working on a computer surfing the Internet, you are considered to be on a client host.

**Router.** Also called a gateway, a router is a machine that connects hosts from one network to another network. For example, if you work in an office environment, the computers within the company can all communicate via the local network created by the administrators. To access the Internet, the computers would have to communicate with a router that would be used to forward network communications to the Internet. Typically when you communicate on a large network (like the Internet), several routers are used before your communication reaches its final destination.

In addition to the networking terms discussed in the last section, there are some additional terms with which you should be familiar. These terms focus more on the different types of networking services that are commonly used, as well as some of the techniques that are used to communicate between machines.

**Packet**. A network packet is used to send network communication between hosts. By breaking down communication into smaller chunks (packets), the data delivery method is much more efficient.

**IP Address**. An Internet Protocol (IP) address is a unique number assigned to a host on a network. Hosts use these numbers to address network communication.

**Mask**. Also called a netmask, subnet mask or mask, a network mask is a number system that can be used to define which IP addresses are considered to be within a single network. Because of how routers perform their functions, networks have to be clearly defined.

**Hostname**. Each host on a network could have its own hostname because names are more natural for humans to remember than numbers, making it easier for us to address network packets to another host. Hostnames are translated into IP addresses before the network packet is sent on the network.

**URL**. A Uniform Resource Locator (URL), also commonly called a web address, is used to locate a resource, like a web page, on the internet. It’s what you type into your web browser to access a web page. For example, http://www.netdevgroup.com. It includes the protocol http:// and the hostname www.netdevgroup.com.

**DHCP**. Hosts can be assigned hostnames, IP addresses and other network-related information by a DHCP (Dynamic Host Configuration Protocol) server. In the world of computers, a protocol is a well-defined set of rules. DHCP defines how network information is assigned to client hosts, and the DHCP server is the machine that provides this information.

**DNS**. As mentioned previously, hostnames are translated into IP addresses, prior to the network packet being sent on the network. So your host needs to know the IP address of all of the other hosts with which you are communicating. When working on a large network (like the Internet), this can pose a challenge as there are so many hosts. A Domain Name System (DNS) provides the service of translating domain names into IP addresses.

**Ethernet**. In a wired network environment, Ethernet is the most common way to physically connect the hosts into a network. Ethernet cables are connected to network cards that support Ethernet connections. Ethernet cables and devices (such as routers) are specifically designed to support different communication speeds, the lowest being 10 Mbps (10 Megabits per second) and the highest being 100 Gbps (100 gigabits per second). The most common speeds are 100 Mbps and 1 Gbps.

**TCP/IP** . The Transmission Control Protocol/Internet Protocol (TCP/IP) is a fancy name for a collection of protocols (remember, protocol = set of rules) that are used to define how network communication should take place between hosts. While it isn't the only collection of protocols used to define network communication, it is the most often utilized one. As an example, TCP/IP includes the definition of how IP addresses and network masks work.

IP Addresses

As previously mentioned, hosts address network packets by using the IP address of the destination machine. The network packet also includes a return address, which is the IP address of the sending machine.

There are, in fact, two different types of IP addresses: IPv4 and IPv6. To understand why there are two different types, you need to understand a brief bit of IP addressing history.

For many years, the IP addressing technique that was used by all computers was IPv4. In an IPv4 address, a total of four 8-bit numbers are used to define the address. This is considered a 32-bit address (4 x 8 = 32). For example:

*192.168.10.120.*

*8-bit refers to numbers from 0 to 255.*

Each host on the Internet must have a unique IP address. In an IPv4 environment, there is a technical limit of about 4.3 billion IP addresses. However, many of these IP addresses are not usable for various reasons. Also, many organizations haven't made use of all of the IP addresses they have available.

While it seems like there should be plenty of IP addresses to go around, various factors have led to a problem: the Internet started running out of IP addresses.

This issue encouraged the development of IPv6. IPv6 was officially created in 1998. In an IPv6 network the addresses are much larger, 128-bit addresses that look like this:

*2001:0db8:85a3:0042:1000:8a2e:0370:7334*

Essentially, this provides for a much larger address pool, so large that running out of addresses any time in the near future is very unlikely.

It is important to note that the difference between IPv4 and IPv6 isn't just a larger address pool. IPv6 has many other advanced features that address some of the limitations of IPv4, including better speed, more advanced package management and more efficient data transportation.

Considering all the advantages, you would think that by now all hosts would be using IPv6. However, the majority of network-attached devices in the world still use IPv4 (something like 98-99% of all devices).

So, why hasn't the world embraced the superior technology of IPv6?

There are primarily two reasons:

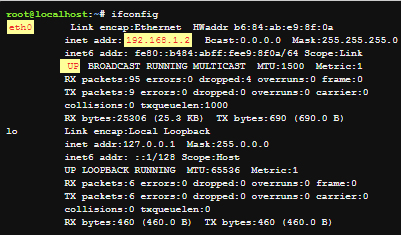
**NAT:** Invented to overcome the possibility of running out of IP addresses in an IPv4 environment, Net Address Translation (NAT) used a technique to provide more hosts access to the Internet. In a nutshell, a group of hosts is placed into a private network with no direct access to the Internet; a special router provides Internet access, and only this one router needs an IP address to communicate on the Internet. In other words, a group of hosts shares a single IP address, meaning a lot more computers can attach to the Internet. This feature means the need to move to IPv6 is less critical than before the invention of NAT.

**Porting:** Porting is switching over from one technology to another. IPv6 has a lot of great new features, but all of the hosts need to be able to utilize these features. Getting everyone on the Internet (or even just some) to make these changes poses a challenge. ‌​

Nonetheless, most experts agree that IPv6 will eventually replace IPv4, so understanding the basics of both is recommended for those who work in the IT industry.

Network Tools

The *ifconfig* command stands for interface configuration and is used to display network configuration information. Not all network settings are covered in this course, but it is important to note from the output below that the IP address of the primary network device eth0 is 192.168.1.2 and that the device is currently active UP:



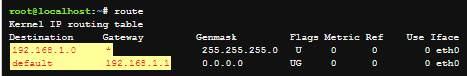
The ifconfig command is becoming obsolete in some Linux distributions (deprecated) and is being replaced with a form of the ip command, specifically ip addr show.

The *ip* command differs from ifconfig in several important manners, chiefly that through its increased functionality and set of options, it can almost be a one-stop shop for configuration and control of a system’s networking. The format for the ip command is as follows:

ip [OPTIONS] OBJECT COMMAND

While ifconfig is limited primarily to modification of networking parameters, and displaying the configuration details of networking components, the ip command branches out to do some of the work of several other legacy commands such as route and arp.

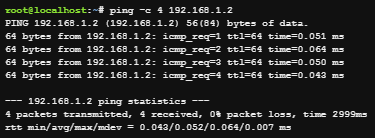
Recall that a router (or gateway) is a machine that allows hosts from one network to communicate with another network. To view a table that describes where network packages are sent, use the *route* command:



The *ping* command can be used to determine if another machine is reachable. If the ping command can send a network package to another machine and receive a response, then you should be able to connect to that machine.

By default, the ping command continues sending packages endlessly. To limit how many pings to send, use the -c option followed by a number indicating how many iterations you desire. The following examples show ping being limited to 4 iterations.

If the ping command is successful, it looks like the following example:



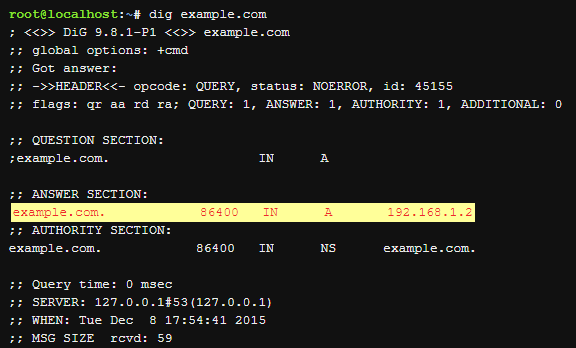
The *netstat* command is a powerful tool that provides a large amount of network information. It can be used to display information about network connections as well as display the routing table similar to the route command.

The *ss* command is designed to show socket statistics and supports all the major packet and socket types. Meant to be a replacement for and to be similar in function to the netstat command, it also shows a lot more information and has more features.

The main reason a user would use the ss command is to view what connections are currently established between their local machine and remote machines, statistics about those connections, etc.

There may be times when you need to test the functionality of the DNS server that your host is using. One way of doing this is to use the dig command, which performs queries on the DNS server to determine if the information needed is available on the server.

In the following example, the *dig* command is used to determine the IP address of the example.com host:



In its simplest form, the *host* command works with DNS to associate a hostname with an IP address. As used in a previous example, example.com is associated with the IP address of 192.168.1.2:



The *ssh* command allows you to connect to another machine across the network, log in and then perform tasks on the remote machine.

If you only provide a machine name or IP address to log into, the ssh command assumes you want to log in using the same username that you are currently logged in as. To use a different username, use the syntax:

*username@hostname*

To return back to the local machine, use the exit command.

**Хід роботи:**

* 1. Початкова робота в CLI-режимі в Linux ОС сімейства Linux:
  2. Запустіть операційну систему Linux Ubuntu. Виконайте вхід в систему та запустіть термінал ***(якщо виконуєте ЛР у 401 ауд.)***.
  3. Запустіть віртуальну машину Ubuntu\_PC ***(якщо виконуєте завдання ЛР через академію netacad)***
  4. Запустіть свою операційну систему сімейства Linux ***(якщо працюєте на власному ПК та її встановили)*** та запустіть термінал.
  5. Опрацюйте всі приклади команд, що представлені у лабораторних роботах курсу ***NDG Linux Essentials - Lab 13: Where Data is Stored*** та ***Lab 14: Network Configuration.*** Створіть таблицю для опису цих команд

| Назва команди | Її призначення та функціональність |
| --- | --- |
| su | Змінюємо поточного користувача на root |
| ls /proc | Переглядаємо вміст системного каталогу **/proc** (для цього потрібні права доступу root) |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Примітка:** **Скріншоти** виконання команд в терміналі можна **не представляти**, достатньо **коротко описати команди в таблиці**.

* 1. Виконайте практичні завдання у терміналі (продемонструйте скріншоти):
* в даній лабораторній роботі використовувалась команда *cat*, дослідіть її можливості та опишіть для яких задач вона призначена;

Its main purpose is to output the contents of files to standard output.

Viewing File Contents

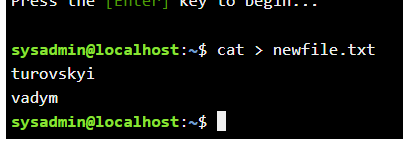
Creating Files

Redirecting Output

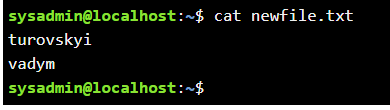
Merging Files

* \*продемонструйте приклади, коли команда *cat* використовується для створення файлу, перегляду вмісту файлу, перенаправлення інформації у інший файл, склеювання декількох файлів в один;

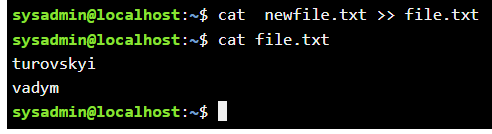
**створення**



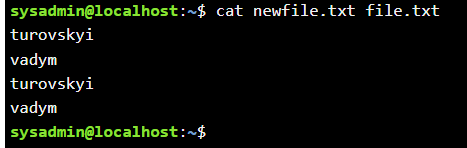
**виведення**



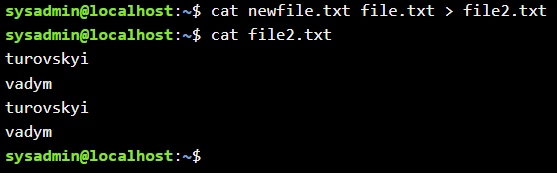
**Перенаправлення в інший файл**



**Склеювання**

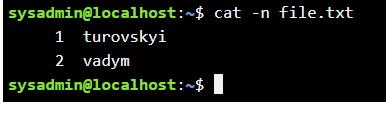


**Склеювання та перенаправлення**

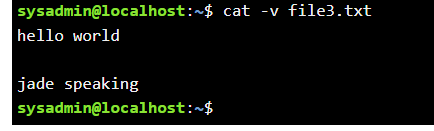


* \*які параметри команди *cat* треба використати, щоб пронумерувати рядки файлу, відобразити недруковані символи, видалити порожні рядки?

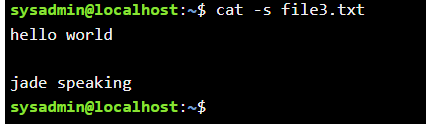
**-n нумерація**



**-v відображення недрукованих символів**



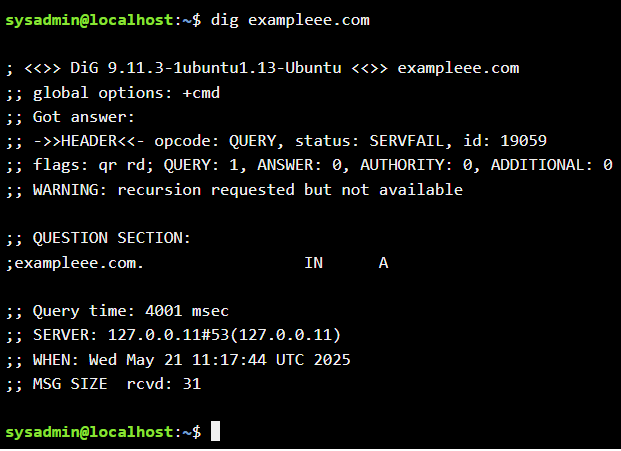
**-s видалення пустих рядків**



**Використовував команду в cisco, чогось не працює**

* \*\*опишіть можливості команди *dig* та наведіть приклади;

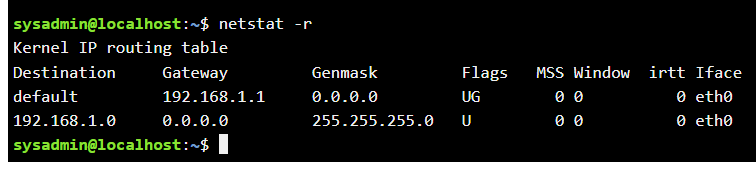
**Obtaining different types of DNS records**



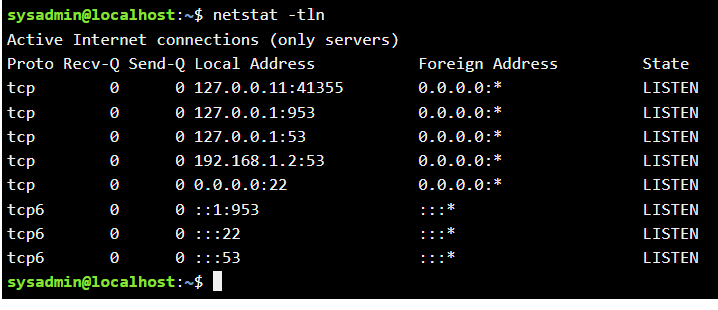


* \*\*опишіть можливості команди *netstat* та наведіть приклади;.

**netstat is used to display information related to network connections, routing tables, interface statistics, masquerade connections, and multicast group membership.**



**The netstat command is also used to display open ports.**



**Висновок:** на практиці розглянув команди *cat* якавикористовується для створення файлу, перегляду вмісту файлу, перенаправлення інформації у інший файл, склеювання декількох файлів в один. Параметри команди *cat* якітреба використати, щоб пронумерувати рядки файлу, відобразити недруковані символи, видалити порожні рядки. Також розглянув dig та netstat.