CS 3307-01 – Operating System 2 – AY2023-T5  
Assignment Unit 3

Instructor: Rupali Memane

University of the People

Jul 2023

An Introduction To File Systems, Prompts, and Remote Procedure Calls

## File systems

In an OS context, a file system is a rule set showing how to name, store, and retrieve files on a partition, a logical division of a physical storage device (Lavarian, 2022). Based on file systems, OS can organize files and directories efficiently, consistently, and securely. Without file systems, there are no files and directories, only contiguous data blocks on a storage device. Thus, to get the expected data blocks, one must know the start and end addresses (sector numbers). We often call storage devices, in this situation, raw disks. In fact, if we don't need file management, we don't need file systems; for example, with embedded OSes, deleted data recovery, and disk backup.

### A typical scenario

Back to the world of OS with file systems, after plugging a physical storage device into a computer, we often do the following steps:

1. Partitioning
   1. Define the scheme and initialize the partition table.
   2. Create a partition and update the partition table.
2. Format the newly created partition with a particular file system.
3. Mount the formatted partition into the file system tree.

#### Partitioning schemes

There are two popular partitioning schemes, Master Boot Record (MBR) and GUID Partition Table (GPT). The latter is more common today due to the limits of the first, such as the maximum size of a partition and no backup for the MBR sector (Lavarian, 2022).

#### Partition creation

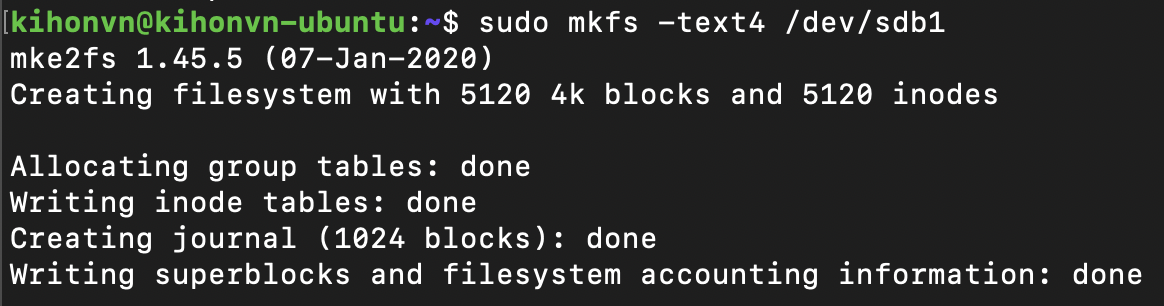
When creating a partition, the required information includes its number, first sector, and size.

Although we can use commands in a shell to accomplish these steps, working with a GUI tool whenever possible is a better approach since it reduces the underline complexity and error-prone typing. Unfortunately, most server-version Linux distros don't provide GUI. Therefore, if you tend to become a system administrator or a DevOp, looking at `fdisk(8)` and `mkfs(8)` may help you.

A screen shot of a computer

Description automatically generated

*Figure 1 – Using fdisk(8) command to create a new partition on Ubuntu Server OS*



*Figure 2 – Using mkfs(8) command to format a partition on Ubuntu Server OS*

Of course, this is only for the first time (or you want to practice it). In many cases, the manufacturers already did all these for us. Now we can create, write, copy, modify, execute, and delete our files and directories on mounted partitions.

## Prompts

In a command-line interface, when the computer waits for user input, it displays a prompt before the cursor. A default representation of a prompt often includes:

* Current username
* Computer's name
* Current directory name/path
* A prompt character: `%` (macOS) and `$` (Linux)

For example, a prompt of the `kihonvn` user on the `kihonvn-ubuntu` computer at the user home directory can be:

In Linux, `kihonvn@kihonvn-ubuntu:~$`

In macOS, `kihonvn@kihonvn-ubuntu ~ %`

## Remote Procedure Call (RPC)

RPC is a protocol designed for calling procedures of processes running on other hosts (of course, it also works well when they are on the same machine). Thus, it promotes client-server architecture and distributed programming.

We can describe the steps of a typical RPC flow as the following:

* On the client side, the client code creates/gets the RPC client stub and calls the procedure.
* Based on the predefined transport protocol information, the RPC client stub builds the request message (including marshaling parameters and setting them into the request message), then sends it through the transport protocol and waits for the server response.
* The RPC server receives the message from the transport protocol and delivers the message to the corresponding RPC server stub. The RPC server stub demarshalls the received message to get the given arguments and call the corresponding function with them.
* When the function completes and returns, the RPC server stub builds the response message (including marshaling the returned value, if any, and setting it into the response message), then sends it back to the client.
* After receiving the server response, the client sends it to the corresponding RPC client stub. The RPC client stub demarshals the response message and sends the returned value to the caller code.

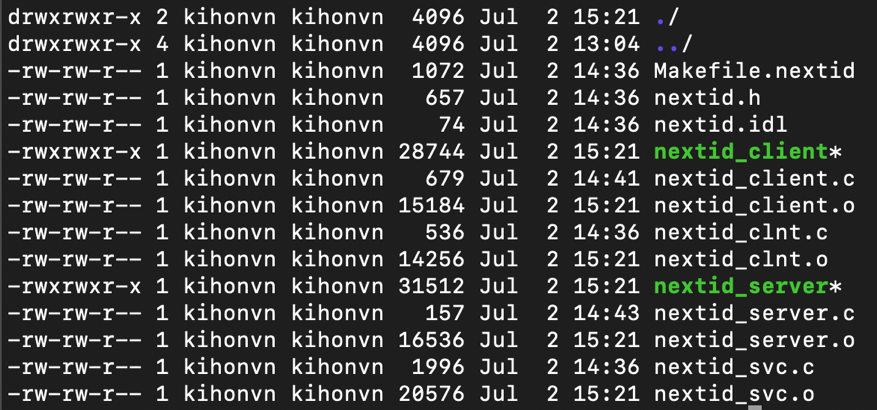
Although RPC protocol lets the implementers freely choose the transport protocol, most RPC frameworks use TCP (Transmission Control Protocol) or UDP (User Datagram Control) to implement it. If you know the IDL and C language and want to try RPC, look at RPC v2 RFC of Sun Microsystem (1988) and `rpcgen(1)` and `make(1)` commands.

## 

*Figure 3 – nextid.idl – A RPC definition example*



*Figure 4 – The output after using rpcgen(1) on nextid.idl.*



*Figure 5 – After using make(1) on Makefile.nextid, we have executable client and server files*

## Security in Unix/Linux

At the OS level, Unix/Linux provides it through:

* File permission (Discretionary Access Control - DAC)
* Process isolation
* Mandatory Access Control (MAC)
* Firewall (iptables)
* Security auditing
* Security patches

(Red Hat, Inc, 2023; Tevault, 2018)

In addition, due to the large community, Unix/Linux has a lot of security extensions that help enhance the system's security.

References

Lavarian, R. (2022, January 11). What Is a File System? Types of Computer File Systems and How they Work – Explained with Examples. *freeCodeCamp.org* https://www.freecodecamp.org/news/file-systems-architecture-explained/

Sun Microsystems, Inc. (1988). RPC: Remote Procedure Call Protocol Specification Version 2. <https://datatracker.ietf.org/doc/html/rfc1057>

Red Hat, Inc. (2023). *Red Hat Enterprise Linux 8 Security hardening*. <https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/pdf/security_hardening/red_hat_enterprise_linux-8-security_hardening-en-us.pdf>

Tevault, A. D. (2018). *Mastering Linux Security and Hardening*. Packt Publishing. https://linuxbg.eu/books/w\_pacb92.pdf