Information Technology Institute

Introduction to Operating Systems

Prepared by:

Zeyad Ashraf

File Systems

Content

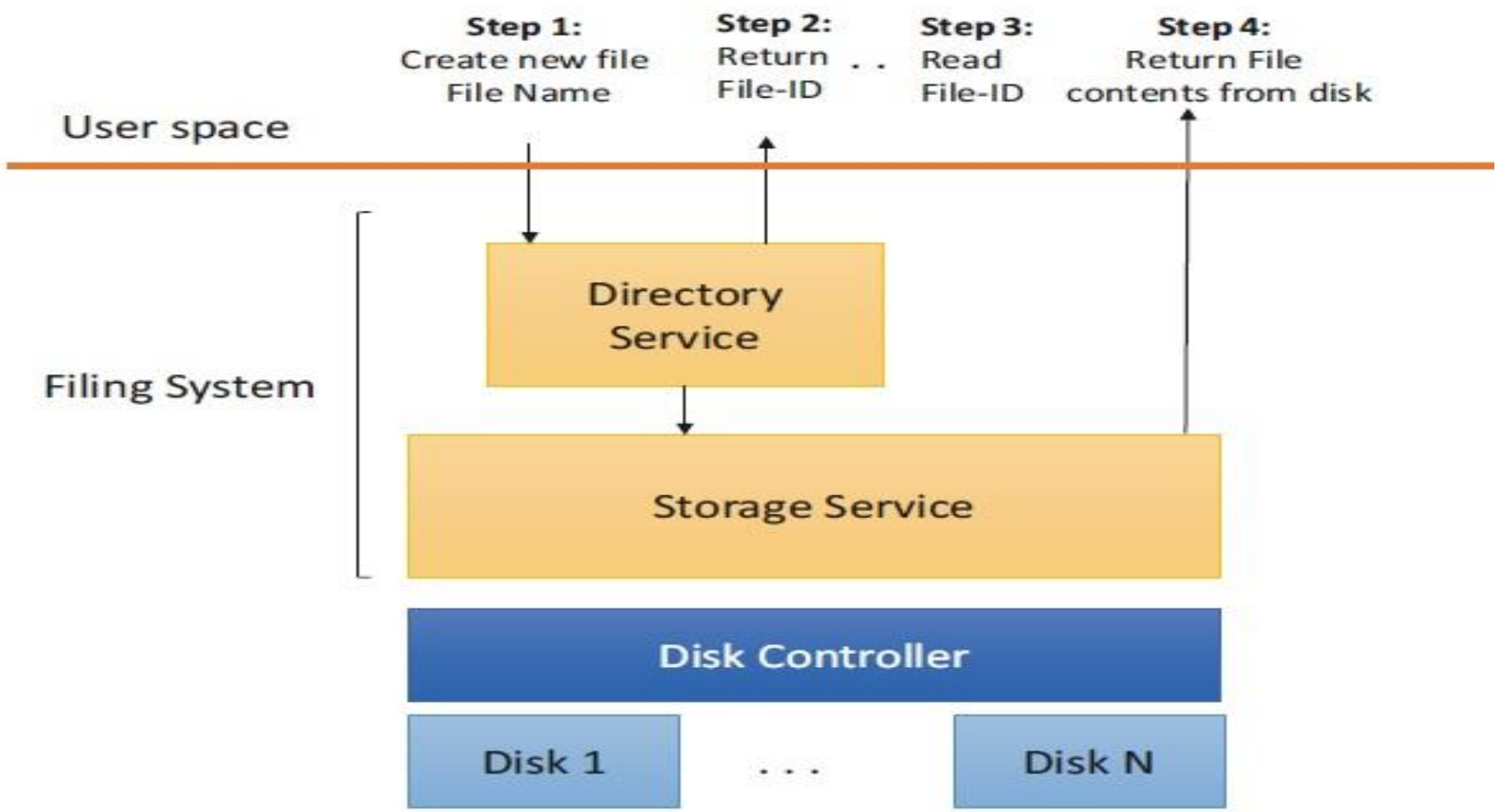
- Need for File Systems
- File Concept
- Directory Name Space
- Access Control
- Concurrency

Course Outlines

5.1 Need for File Systems

- Applications often need to **read and write** files to achieve their goals. We leverage the **OS** to create, read, and write such files on the system.
- We depend on the OS to maintain and manage files on the system.
- OS file systems have two main components to facilitate file management:
  - ***Directory Service***: There is a need to uniquely manage files in a **structured manner**, **manage access**, and provide **Read-Write-Edit** controls on the file system. This is taken care by a layer called as the directory service.
  - ***Storage Service***: There is a need to communicate to the **underlying hardware** such as the disk. This is managed by a storage service that **abstracts** different types of storage devices on the system.

5.1 Need for File Systems



5.2 File Concept

- From the perspective of the user, a file is a collection of related data that is stored together and can be accessed using a unique file ID usually referred as the file name.
- These files can be represented internally by different methods. For example, there could be **.bin** files in Windows, **which only represent a sequence of bytes.**
- There could be other structured contents with headers and specific sections in the file. For example, an **EXE** is also a file format in Windows with specific headers, a body, and controls in place.
- There are also many application-specific files, with their own formats. It is up to the programmer to define and identify if they require a custom file format for their application or if they can leverage a standard or common file format such as the JavaScript Object Notation (**JSON**) or the Extensible Markup Language (**XML**).

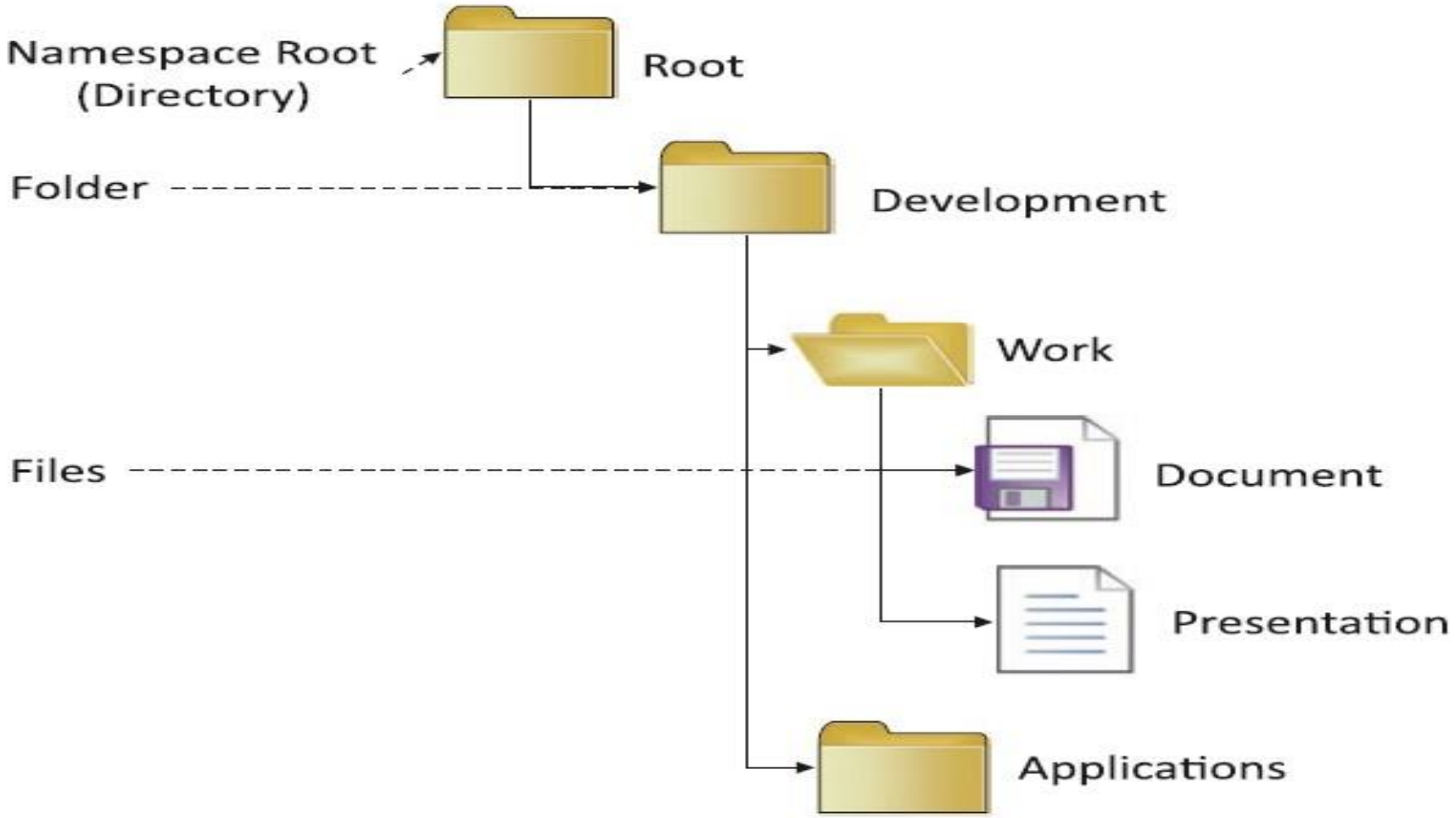
5.2 File Concept

- As a programmer, it may be important to know the **attributes** of the file before accessing it.
- The common attributes of any file include the ***location*** of the file, file ***extension***, ***size***, ***access controls***, and some ***history of operations*** done on the file, to name a few.
- Some of these are part of the so-called ***file control block***, which a user has access to via the OS.
- Most OSs expose **APIs** using which the programmer can access the details in the file control block.
- For the user, these are exposed on the ***graphical user interface via built-in tools*** shipped with the OS.

5.3 Directory Name Space

- Most OSs ***organize*** their files in a ***hierarchical*** form with files organized inside ***folders***.
- Each folder in this case is a directory. This structure is called as the ***directory namespace***.
- The directory service and **namespace** have additional capabilities such as searches by size, type, access levels, and so on.
- The directory namespaces can be ***multileveled*** and adaptive in modern OSs as we can see in the following folder structure with folders created inside another folder

5.3 Directory Name Space



5.4 Access Control

- There are **different access levels** that can be applied at file and **directory** levels.
- The OS provides different access control ***IDs*** and ***permissions*** to different users on the system.
- Also, each file may also have ***different levels of permissions*** to Read, Write, Modify, and so on.
- For example, there may be specific files that we may want anyone to be able to access and Read but not Write and Modify.
- The file system provides and manages the controls to all files when accessed at runtime.
- These may also be ***helpful*** when ***more than one user*** is using the same system.

5.5 Concurrency and Cleanup Control

- There are many cases when the OS needs to ***ensure that a file is not moved or deleted when it is in use***.
- For example, if a user is making changes to a file, the OS needs to ensure that the same file cannot be moved or deleted by another application or process. In this case, the OS would cause the attempt to move or delete the file to fail with an appropriate error code.
- *As a programmer, it is appropriate to access a file with the required* ***access level*** *and* ***mode*** *(Read/Write). This also helps to be in line with the concurrency needs of the OS and guards against inconsistent updates.*

5.5 Concurrency and Cleanup Control

- The OS needs to be able to ***periodically clear temporarily created files*** that may no longer be required for the functioning of the system.
- This is typically done using a ***garbage collector on the system***.
- Many OSs ***mark unused files over a period of time*** and have additional settings that are exposed, which the user can set to clean up files from specified locations automatically.

Access and Protection

Content

- Access and Protection
  - User Mode and Kernel Mode (Rings)

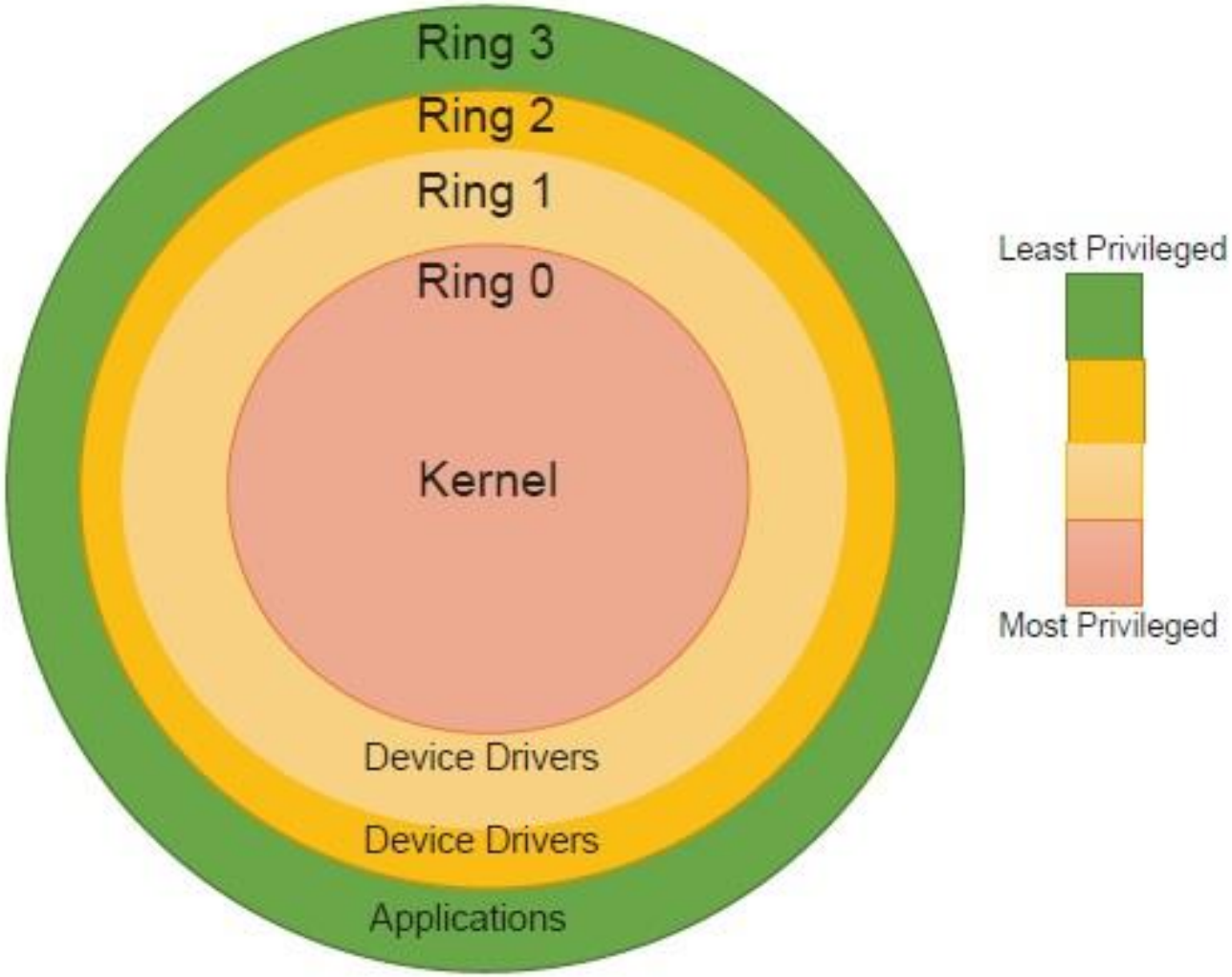
6.1 Access and Protection

- If we have a system that is used by only one user without any access, networked or otherwise, to other systems, there may still not be **assurance** that the contents in the system are protected.
- There is still a need to protect the ***program resources from other applications***.
- Also, there may be a need to protect ***critical devices*** on the system.
- There is always a need to connect and ***share resources and data between systems***.
- The OS provides APIs that help with access control and protection.

6.2 User Mode and Kernel Mode (Rings)

- One of the reasons the separation between user mode and kernel mode is implemented by most OSs is that it ensures ***different privilege levels are granted to programs, based on which mode they run in***.
- OS divides the program execution privileges into ***different rings***.
- Internally, programs running in ***specific rings are associated with specific access levels and privileges***.
- For example, ***applications and user-mode*** services running in *Ring 3* would ***not be able to access the hardware directly***. The ***drivers*** running on the *Ring 0* level would have the ***highest privileges and access to the hardware*** on the system.
- In practice, most OSs only leverage two rings, which are Ring 0 and Ring 3.

6.2 User Mode and Kernel Mode (Rings)



Virtualization and User Interface and Shells

Content

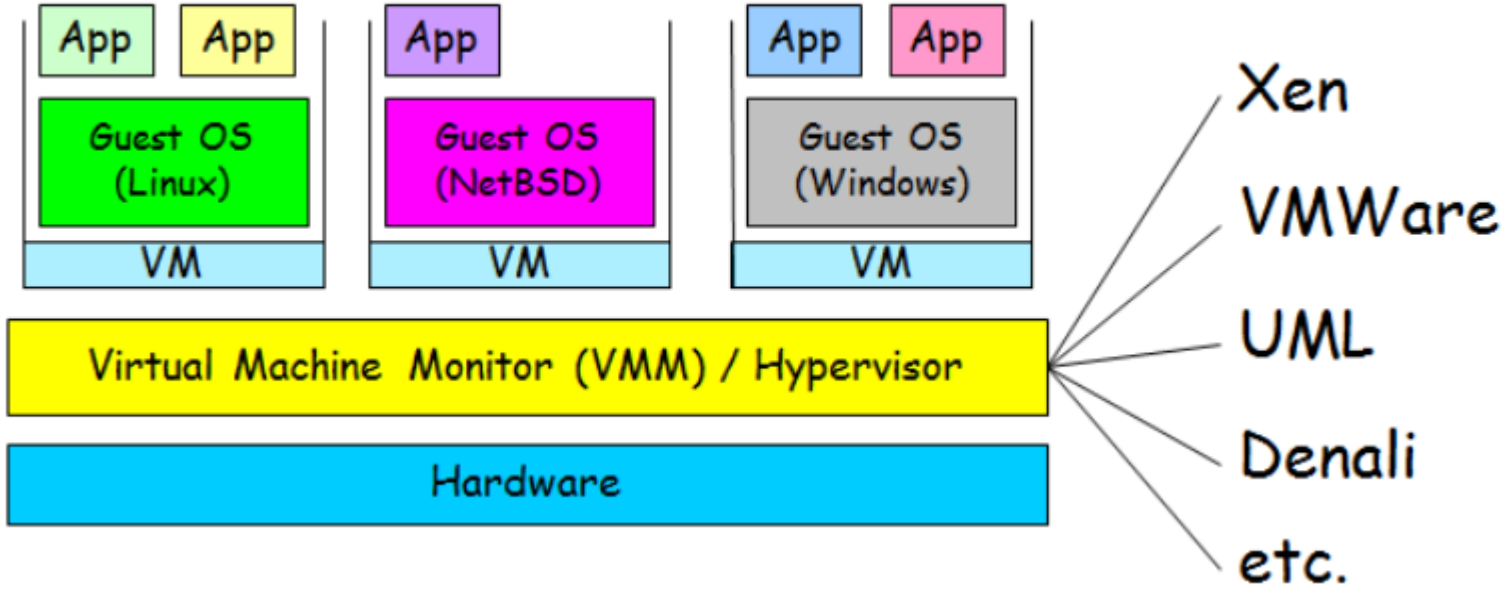
- Virtualization
- Protection
- User Interface and Shell

7.1 Virtualization

- Operating systems and modern hardware provide a feature called virtualization that ***virtualizes the hardware*** such that each calling environment believes it has the dedicated access it needs to function.
- Virtualization is delivered via so-called ***virtual machines*** (VMs).
- A VM has its own ***guest OS***, which may be the same as or different from the underlying ***host OS***.
- The ***host OS*** provides a ***hypervisor***, which manages the access to the hardware.
- The guest OS is usually unaware of the internals and passes any resource/hardware requests to the host OS.
- The user can completely ***customize their VM and perform all their actions*** on this VM ***without affecting the host OS*** or any other VM on the system.

7.1 Virtualization

- VM technology allows multiple virtual machines to run on a single physical machine



7.1 Virtualization

- At a high level, VMs help effectively **utilize** the hardware resources and are used heavily in server and cloud deployments.
- Advantages of virtualization:
  - Run operating systems where the ***physical hardware is unavailable***.
  - Easier to create ***new machines***, ***backup machines***, etc.
  - ***Software testing*** using "clean" installs of operating systems and software.
  - ***Emulate more machines*** than are physically available.
  - ***Debug problems*** (suspend and resume the problem machine).
  - ***Easy migration of virtual machines*** (shutdown needed or not).
  - Run ***legacy systems***

7.2 Protection

- There could be different ***security threats*** that may arise during the usage of a computer.
- A threat could be any ***local or remote program*** that may be attempting to ***compromise the integrity of the resources*** in the system.
- The most common protection would be to ***authenticate the requester*** and apply ***authorization to any new request*** to the system.
- For example, when a request is made to a critical resource, the operating system would verify the ***user*** request (which is called as ***authentication***) and their approved ***access levels*** (which is called ***authorization***) and controls before providing access to a critical resource on the system.

7.2 Protection

- The OS may also have ***Access Control Lists (ACLs)*** that contain mapping of system resources to different permission levels.
- This is used internally before the OS grants permissions to any resource.
- Additionally, the OS may also provide services to encrypt and verify certificates that help with enhancing the security and protection of the system itself.

- *The programmer needs **to be aware of the various access controls** and protection mechanisms in place and use the right protocols and OS services to successfully access resources on the system.*

7.3 User Interface and Shell

- Although the ***user interface*** (UI) is **not** part of the OS kernel itself, this is typically considered to be ***an integral part of the OS***.
- There can be multiple user interfaces for the OS all being implemented either as a ***text-based*** interface or a ***graphical-based*** interface

1. The **graphical user interface** is the rich set of graphical front-end interfaces and functionalities provided by the OS for the user to interact with the computer.
2. There could be an alternate simpler interface through a ***command line shell interface*** that most OSs also provide for communication. This is a text-based interface.

- *It is common for programmers to use the **shell** interface instead of the GUI for quickly traversing through the file system and interacting with the OS.*

7.3 User Interface and Shell

- It is important for the software developer to be aware that the user interface and the shell interface may have an impact on **their choice of programing language**, **handling of command line arguments**, **handling of the standard input-output pipes and interfacing** with OS policies, and so on.
- Note that the user interface and the features can be quite varied and different from each OS to another

Thank You

With My Best Wishes

Zeyad ashraf