Haute école d'ingénierie et d'architecture Fribourg
Hochschule für Technik und Architektur Freiburg

Systèmes d'exploitation

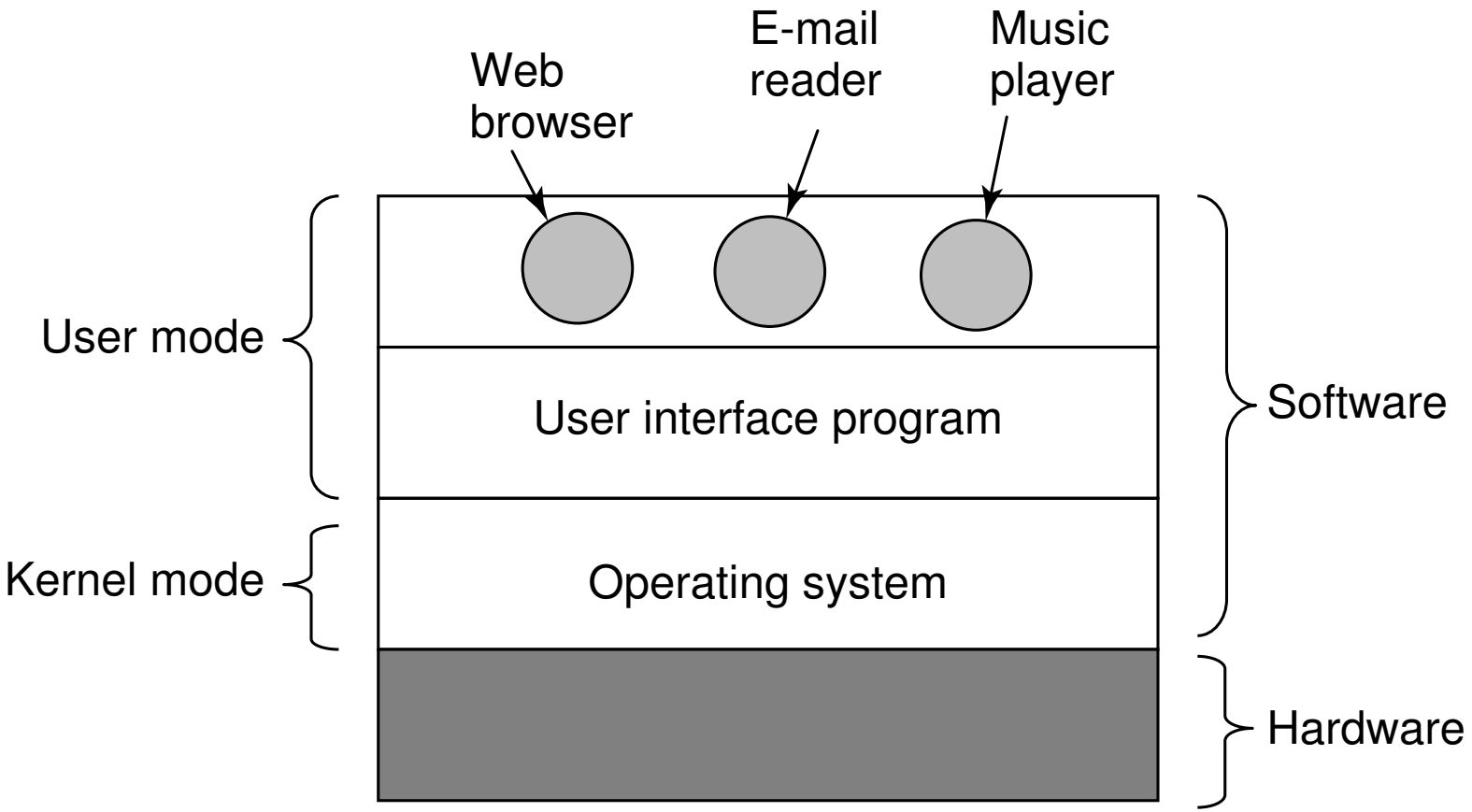
Concepts de bases

Jacques Supcik | 2018/2019 | T-2adfg

I02-handouts | 2018-09-20

Hes·so
Haute Ecole Spécialisée
de Suisse occidentale
Fachhochschule Westschweiz

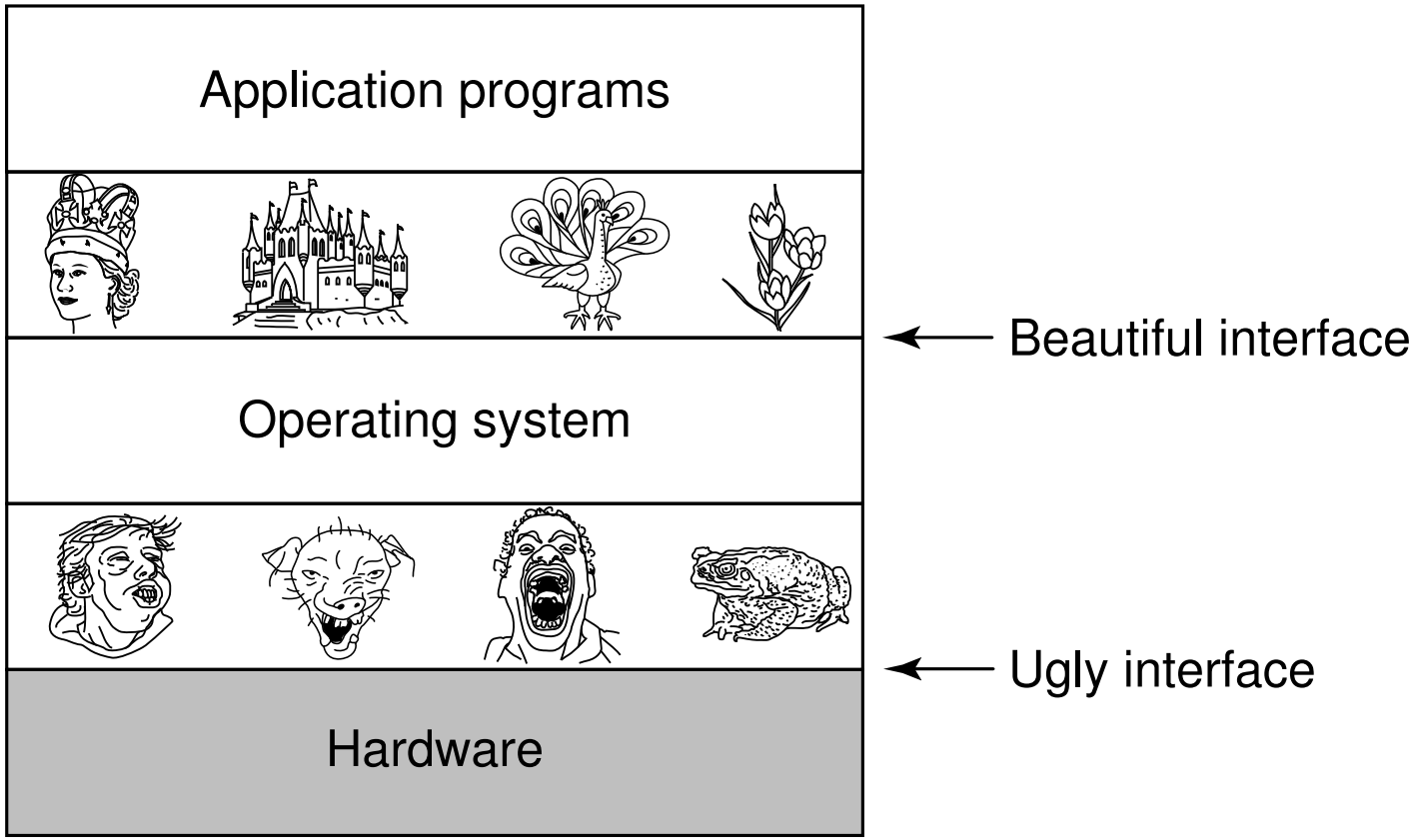
Place du système d'exploitation



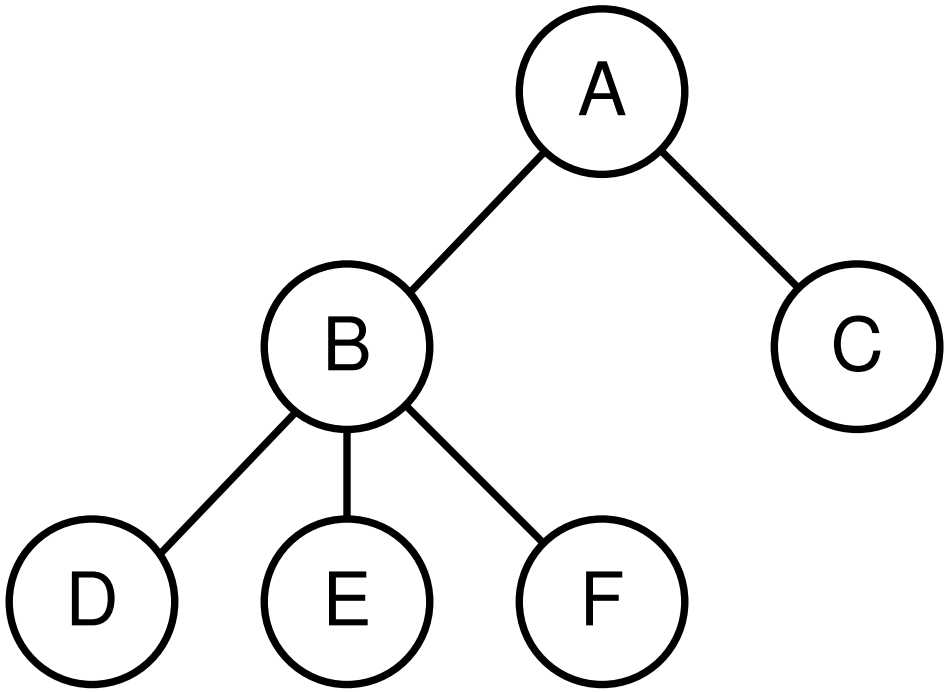
Qu'est-ce qu'un système d'exploitation

- Beaucoup d'utilisateurs ne voient que la GUI «Graphical User Interface» d'un système d'exploitation. Dans ce cours, nous allons nous intéresser à ce qui se passe dans les «entrailles» de la machine.
- Un système d'exploitation sert à gérer les ressources d'un ordinateur :
  - Le ou les processeurs
  - La mémoire
  - Les disques et les fichiers
  - Les interfaces réseau (Ethernet)
  - Le clavier et la souris
  - Le terminal

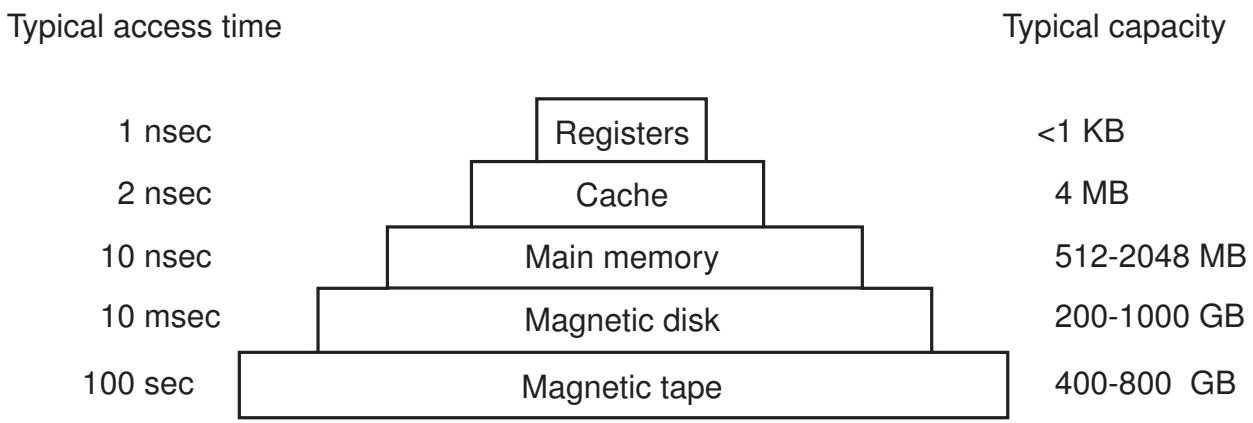
Abstraction



Le système d'exploitation simplifie la vie des programmeurs en leur offrant une interface simplifiée (abstraction) du matériel.



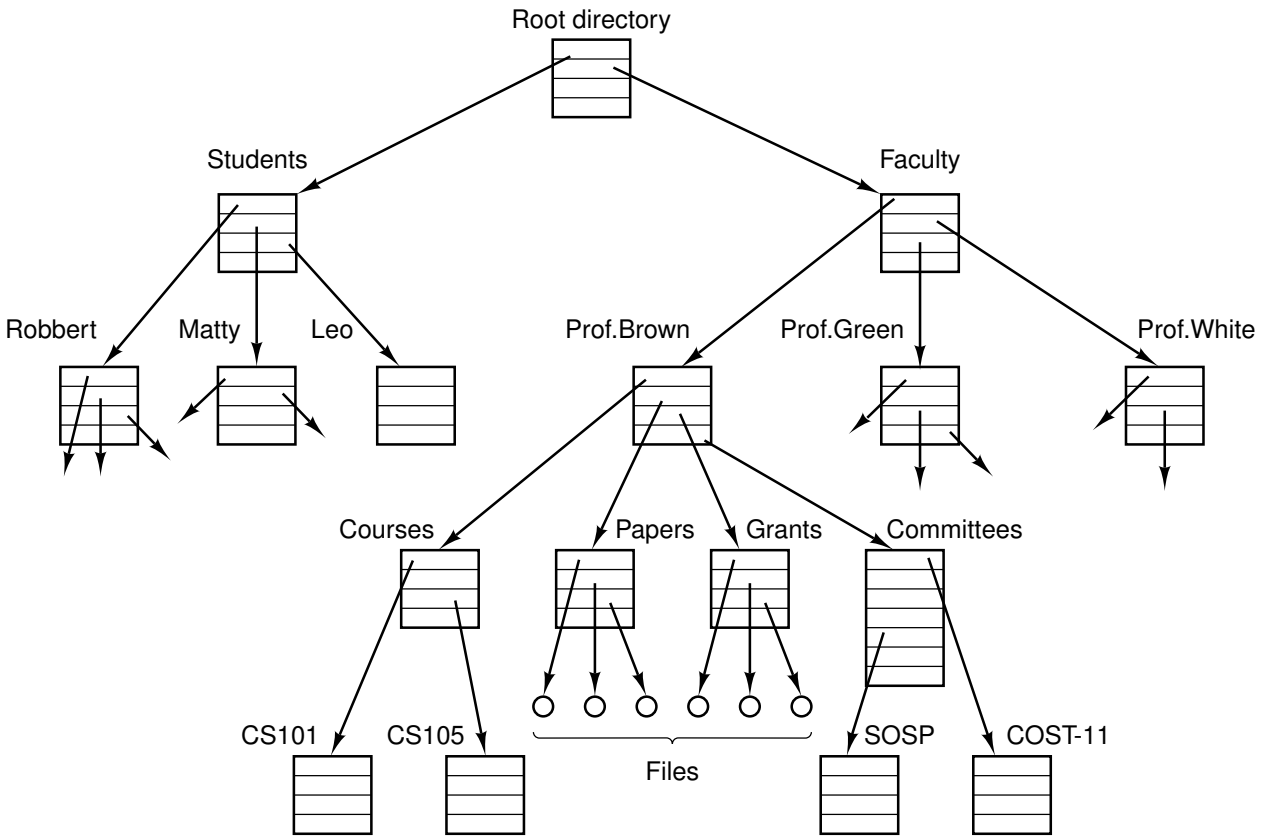
L'arbre des processus. Le processus A a créé les deux processus B et C. Le processus B a créé les processus D, E et F.



- Les disques SSD offrent un «access time» de l'ordre de 100ns et une capacité entre 128 GByte et 1000 GByte.
- Aujourd'hui, la mémoire centrale d'un serveur peut facilement aller jusqu'à 32 GByte. Les disques magnétiques offrent des capacités jusqu'à 4000 GByte.
- Concepts de mémoire virtuelle, d'espace d'adressage (address space).

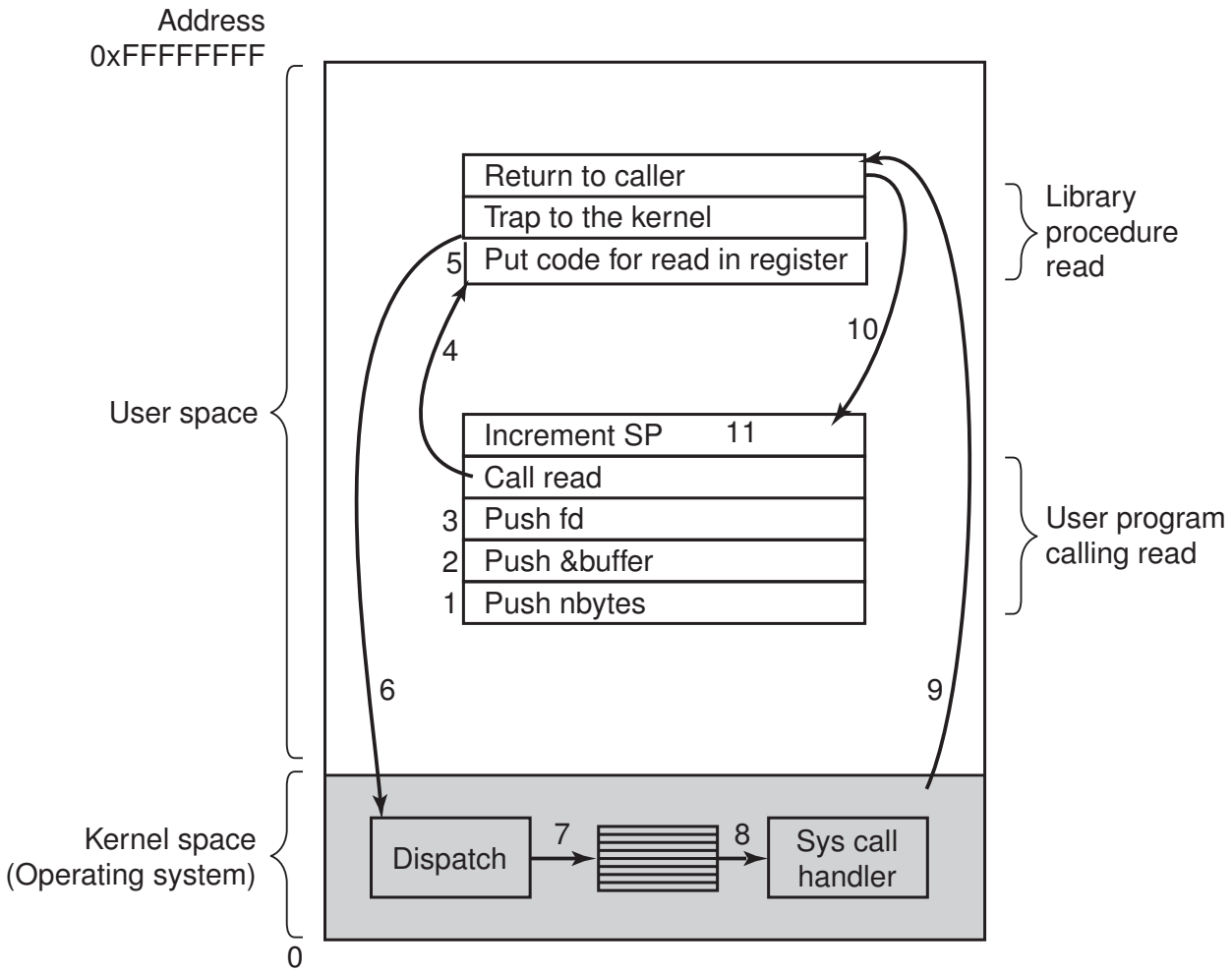
Concepts

Les Fichiers et les répertoires



Les appels systèmes (system calls)

`read(fd, buffer, nbytes)`



Les appels système (system calls)

Gestion des processus

Process management

Call	Description
pid = fork( )	Create a child process identical to the parent
pid = waitpid(pid, &statloc, options)	Wait for a child to terminate
s = execve(name, argv, environp)	Replace a process' core image
exit(status)	Terminate process execution and return status

Les appels système (system calls)

Gestion des fichiers

File management

Call	Description
fd = open(file, how, ...)	Open a file for reading, writing, or both
s = close(fd)	Close an open file
n = read(fd, buffer, nbytes)	Read data from a file into a buffer
n = write(fd, buffer, nbytes)	Write data from a buffer into a file
position = lseek(fd, offset, whence)	Move the file pointer
s = stat(name, &buf)	Get a file's status information

Les appels système (system calls)

Gestion des répertoires et du système de fichiers

Directory and file system management

Call	Description
s = mkdir(name, mode)	Create a new directory
s = rmdir(name)	Remove an empty directory
s = link(name1, name2)	Create a new entry, name2, pointing to name1
s = unlink(name)	Remove a directory entry
s = mount(special, name, flag)	Mount a file system
s = umount(special)	Unmount a file system

Les appels systèmes (system calls)

Divers

Miscellaneous

Call	Description
s = chdir(dirname)	Change the working directory
s = chmod(name, mode)	Change a file's protection bits
s = kill(pid, signal)	Send a signal to a process
seconds = time(&seconds)	Get the elapsed time since Jan. 1, 1970

La valeur de retour «s» indique si l'opération a bien été effectuée (s = 0) ou non (s = -1).

Le shell

Version très simplifiée

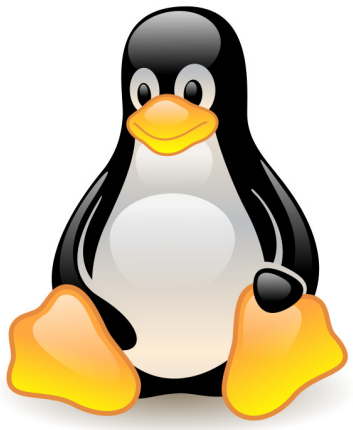
```
#define TRUE 1

while (TRUE) {                                  /* repeat forever */
        type_prompt( );                         /* display prompt on the screen */
        read_command(command, parameters);      /* read input from terminal */

        if (fork( ) != 0) {                     /* fork off child process */
             /* Parent code. */
             waitpid(–1, &status, 0);           /* wait for child to exit */
        } else {
             /* Child code. */
             execve(command, parameters, 0);    /* execute command */
        }
}
```

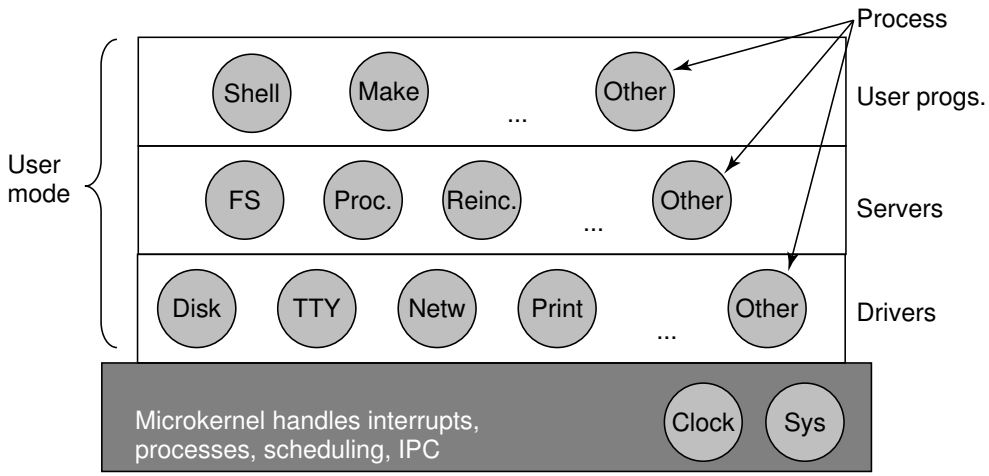
Structure d'un système d'exploitation

- Les systèmes monolithiques
- Les micronoyaux
- Les systèmes en couche (layered system)
- Le modèle «client-serveur»
- Les machines virtuelles



Structure d'un système d'exploitation

- Les systèmes monolithiques
- Les micronoyaux
- Les systèmes en couche (layered system)
- Le modèle «client-serveur»
- Les machines virtuelles



En moyenne, on considère qu'on code contient entre 2 et 10 bugs par 1'000 lignes de code. Le noyau Linux contient plus de 20'000'000 de lignes de code et Windows 10 plus de 50'000'000. En comparaison, MINIX contient moins de 15'000 lignes de code.

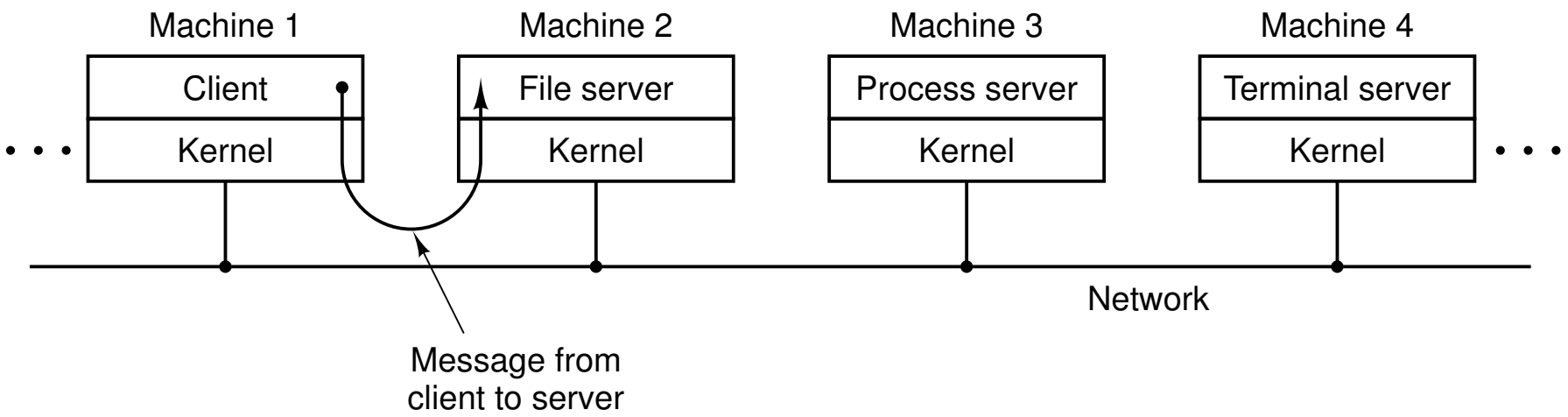
Structure d'un système d'exploitation

- Les systèmes monolithiques
- Les micronoyaux
- Les systèmes en couche (layered system)
- Le modèle «client-serveur»
- Les machines virtuelles

Layer	Function
5	The operator
4	User programs
3	Input/output management
2	Operator-process communication
1	Memory and drum management
0	Processor allocation and multiprogramming

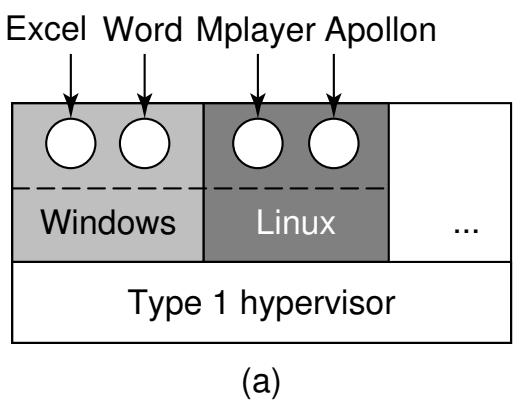
Structure d'un système d'exploitation

- Les systèmes monolithiques
- Les micronoyaux
- Les systèmes en couche (layered system)
- Le modèle «client-serveur»
- Les machines virtuelles

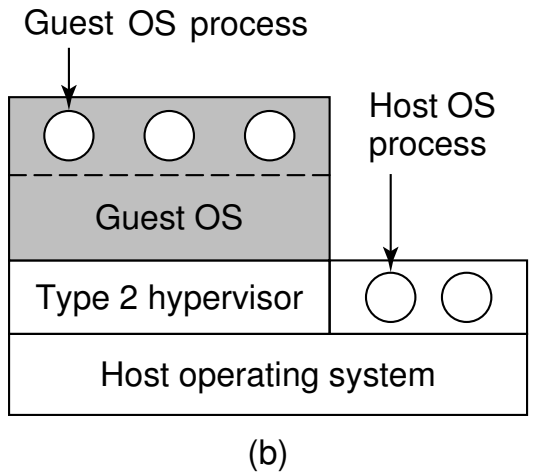


Structure d'un système d'exploitation

- Les systèmes monolithiques
- Les micronoyaux
- Les systèmes en couche (layered system)
- Le modèle «client-serveur»
- Les machines virtuelles



(a)



(b)