

# Les Structures d'un Système d'Exploitation

---

# Menu

---

- Les services d'un système d'exploitation
- L'interface d'un système d'exploitation
- Composants du système d'exploitation
- Démarrage du système informatique

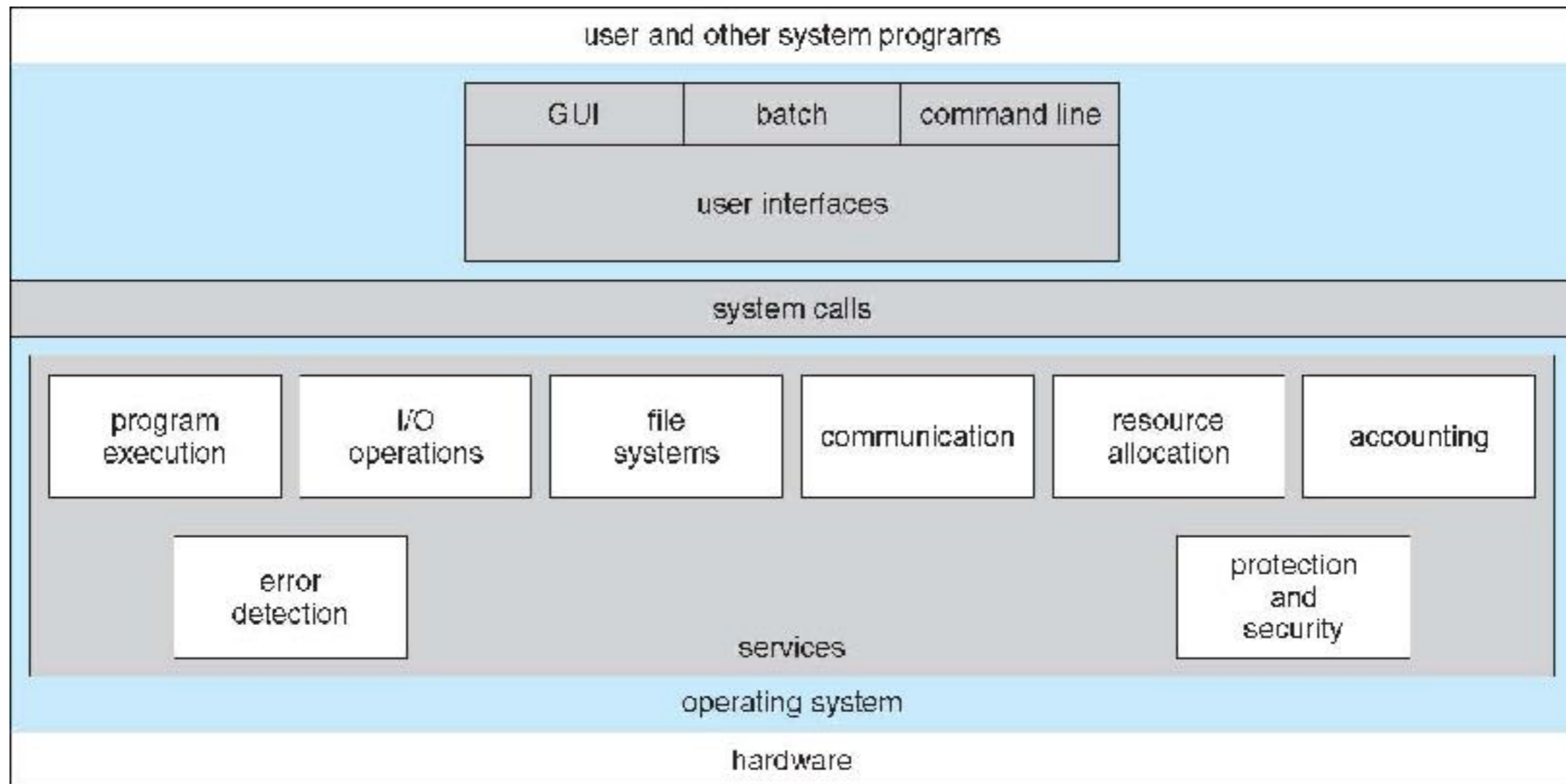
# Menu

---

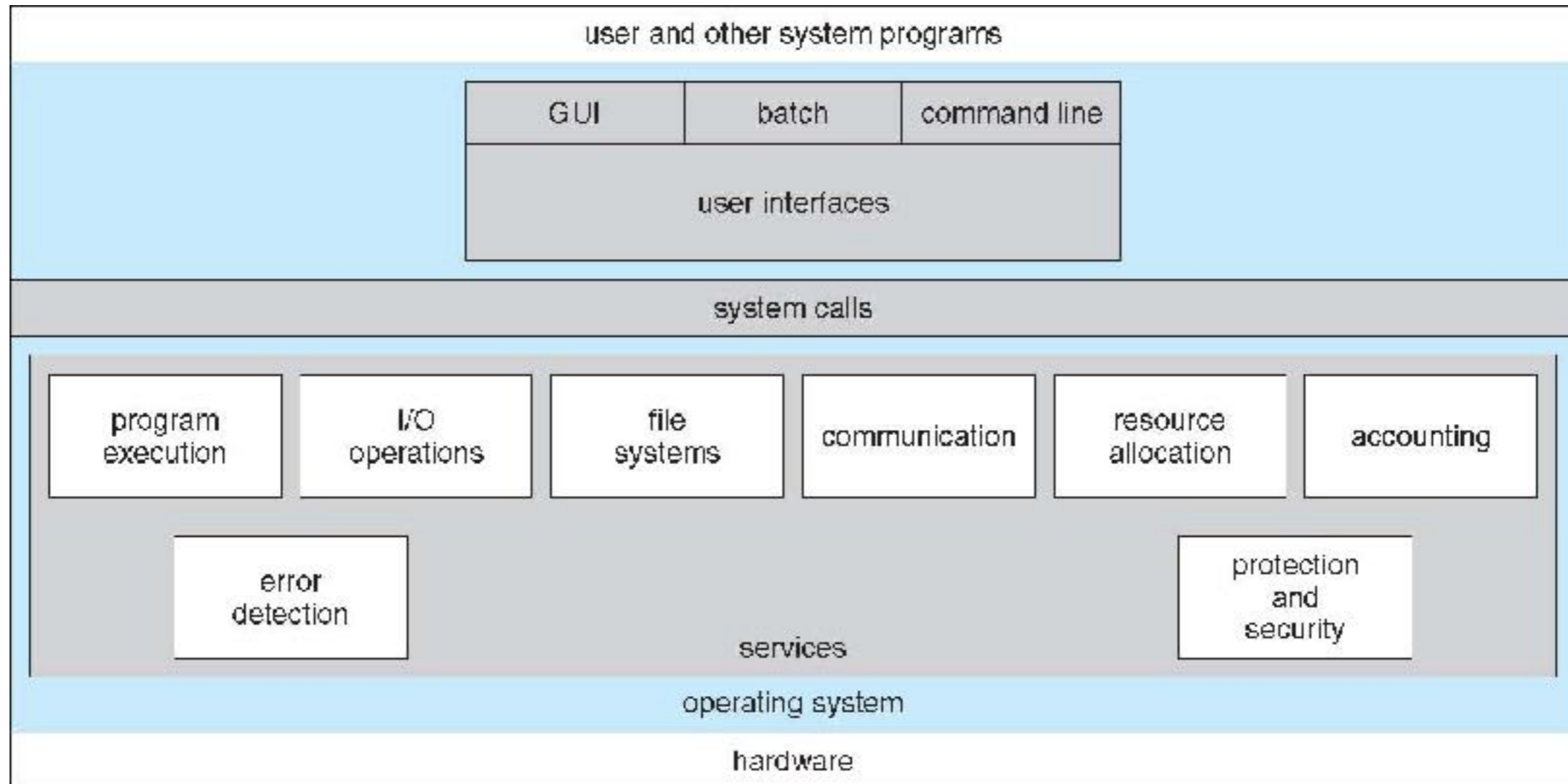
- **Les services d'un système d'exploitation**
- L'interface d'un système d'exploitation
- Composants du système d'exploitation
- Démarrage du système informatique

# Les Services

- Les systèmes d'exploitation fournissent un environnement pour l'exécution de programmes et de services aux programmes et utilisateurs

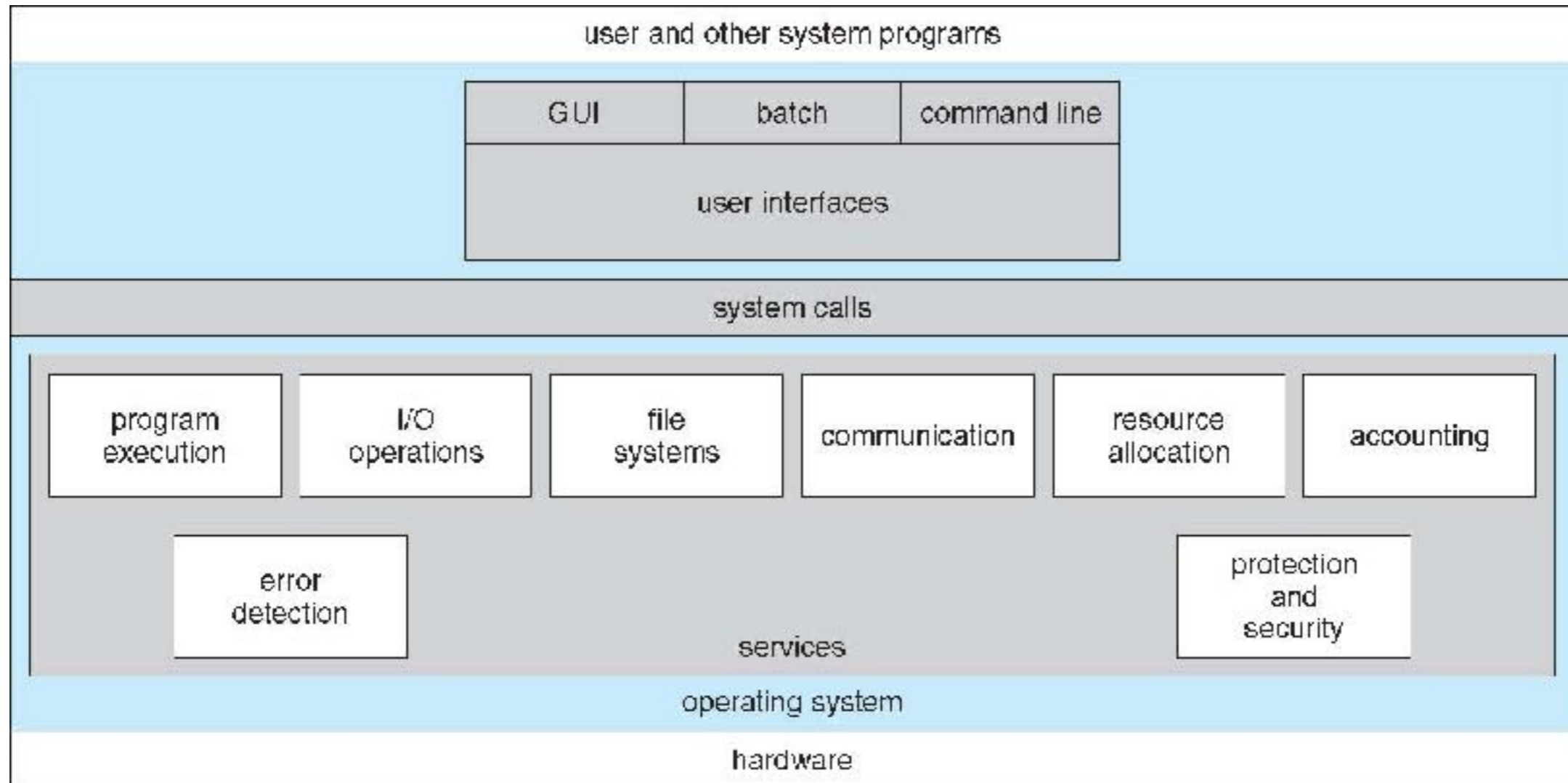


# Les Services (Utilisateur)



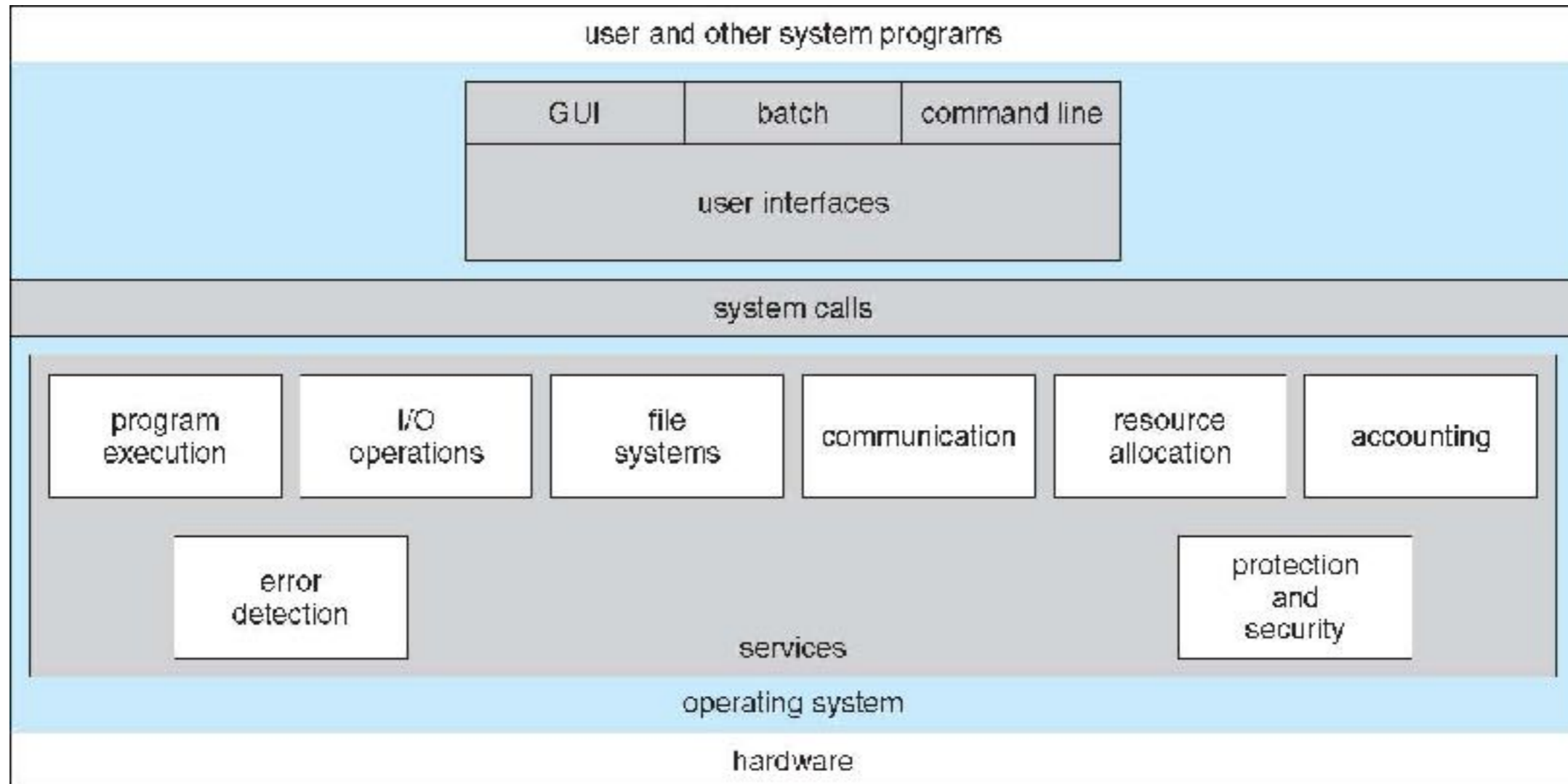
- **Interface utilisateur** - Presque tous les systèmes d'exploitation ont une interface utilisateur (UI).
  - Varie entre **la ligne de commande (CLI)**, **l'interface utilisateur graphique (GUI)**, le batch ou les combinaisons de ceux-ci

# Les Services (Utilisateur)



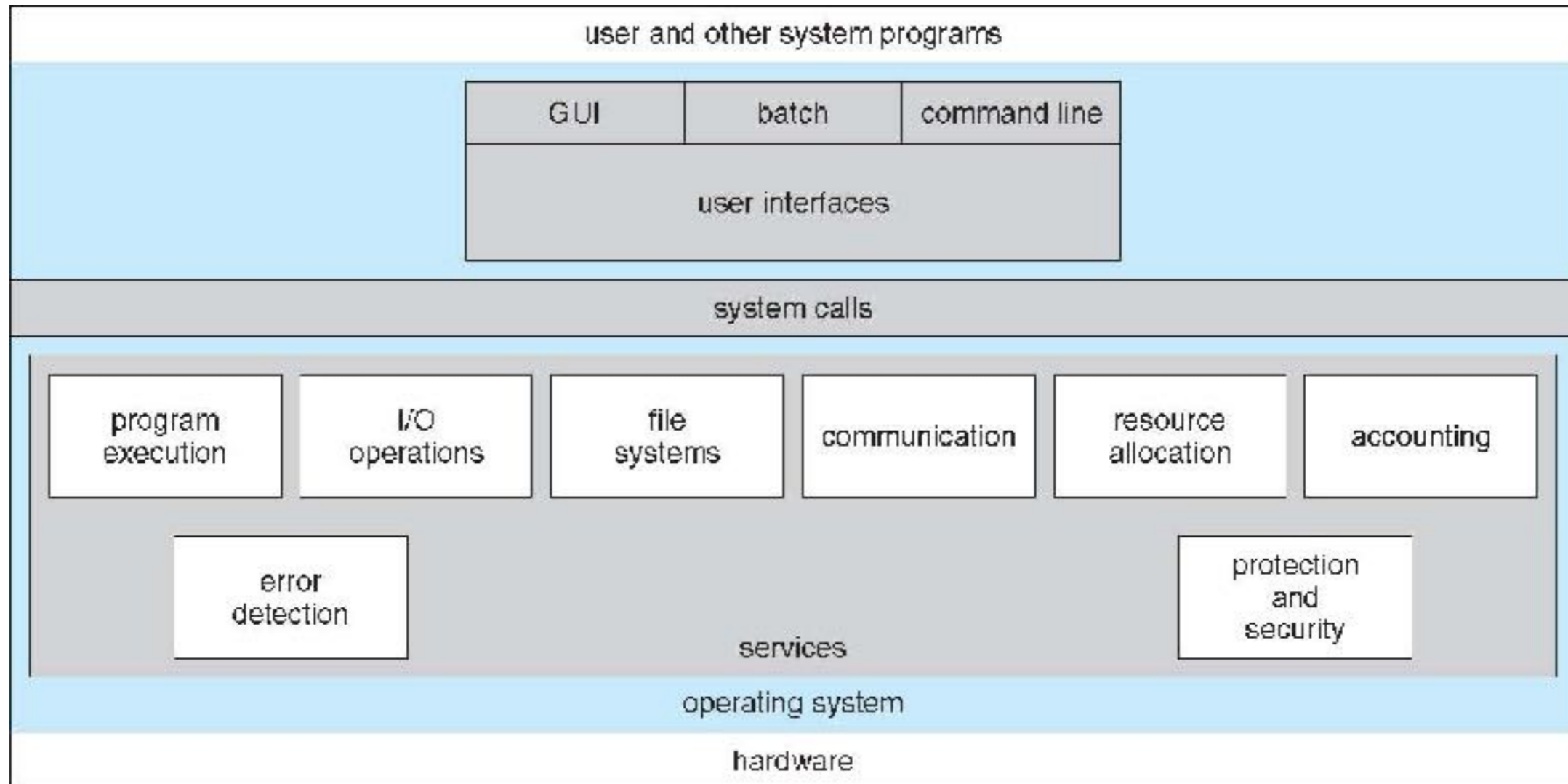
- **Exécution du programme** - Le système doit être capable de charger un programme en mémoire et d'exécuter ce programme, terminer l'exécution, normalement ou anormalement (indiquer une erreur)

# Les Services (Utilisateur)



- **Opérations d'E / S** - Un programme en cours peut nécessiter des E / S, ce qui peut impliquer un fichier ou un périphérique d'E / S

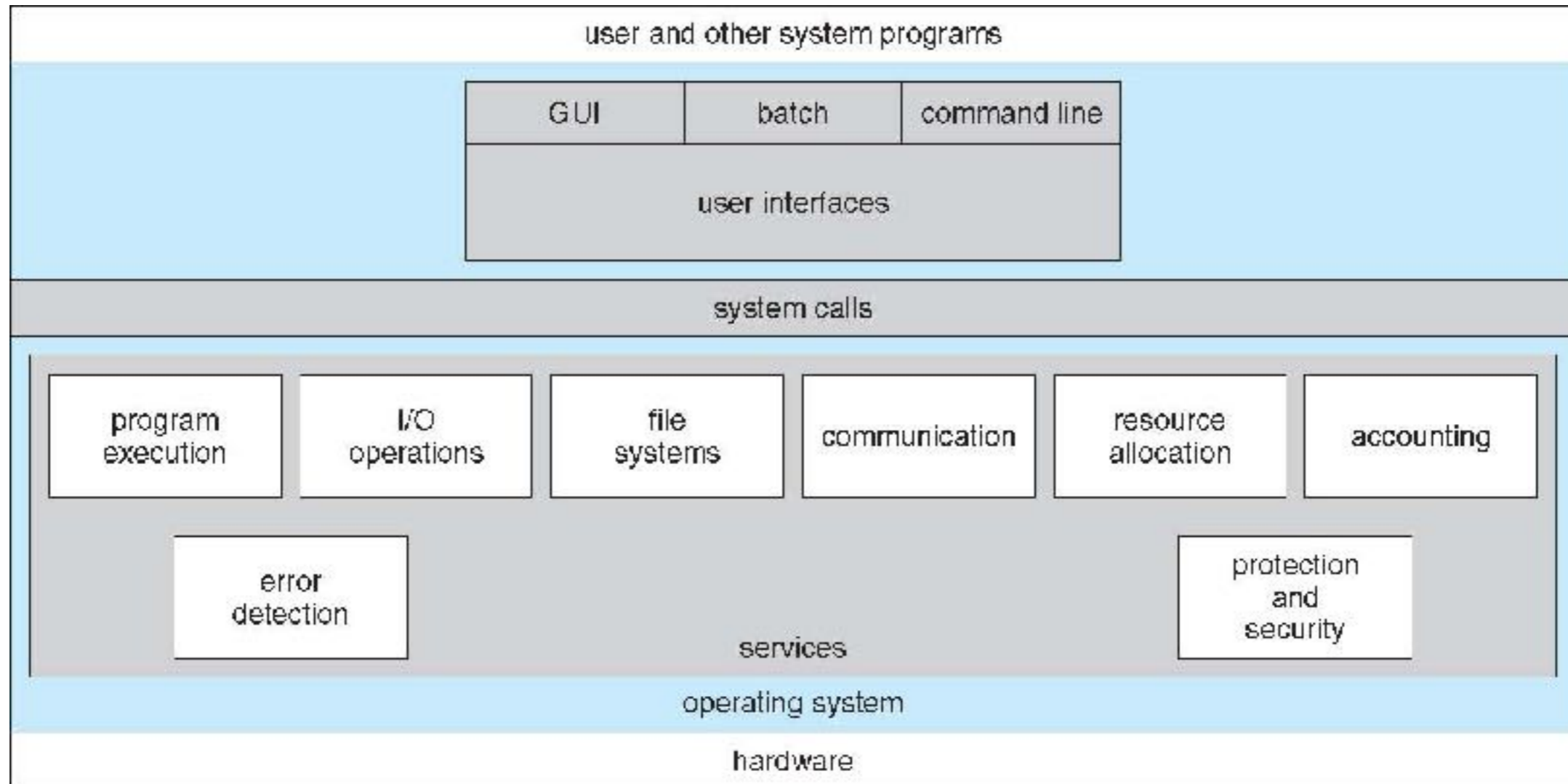
# Les Services (Utilisateur)



- **Manipulation du système de fichiers** - Le système de fichiers présente un intérêt particulier. Les programmes doivent lire et écrire des fichiers et des répertoires, les créer et les supprimer, les rechercher, lister les informations du fichier, gérer les permissions.

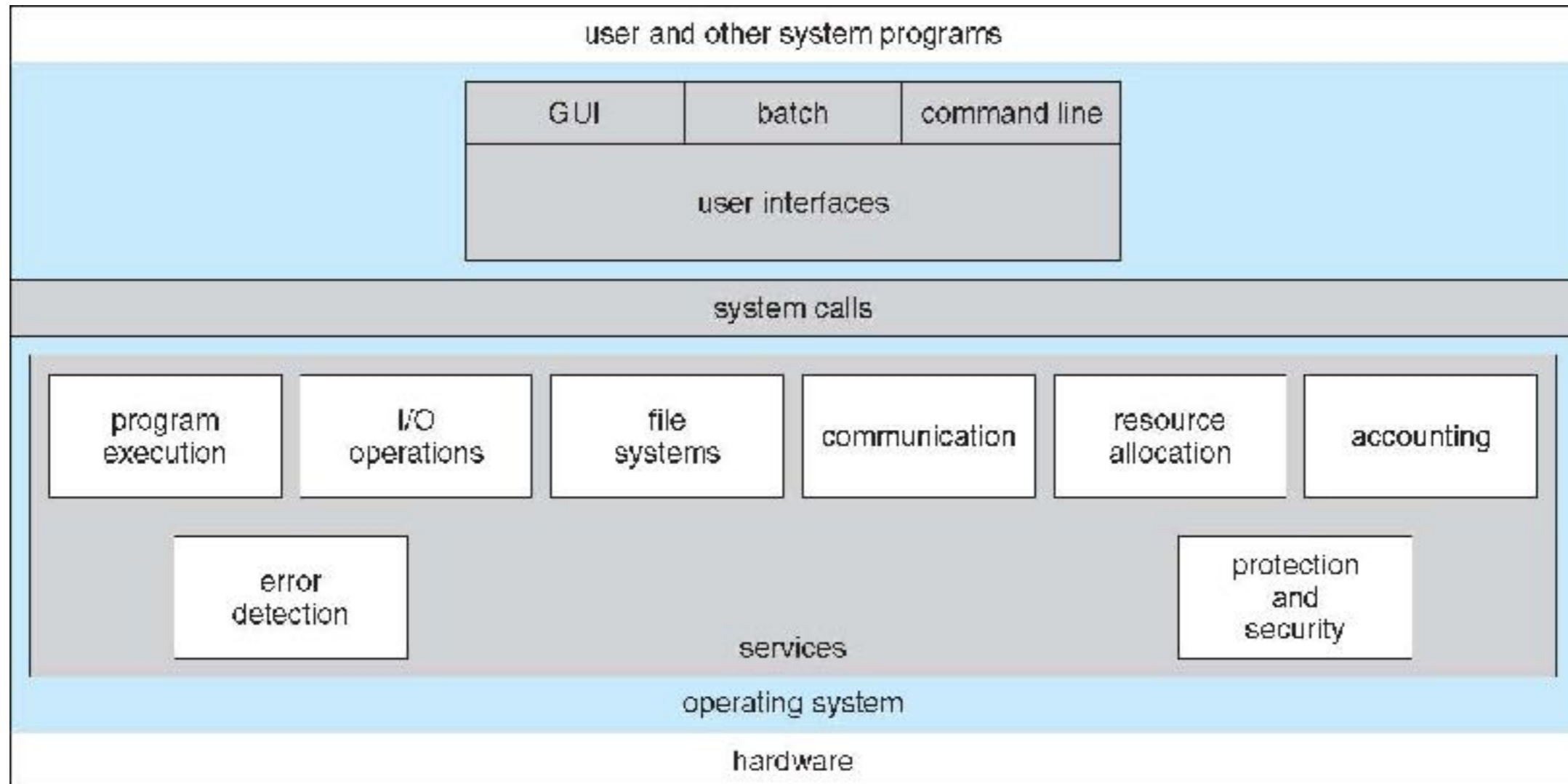


# Les Services (Utilisateur)



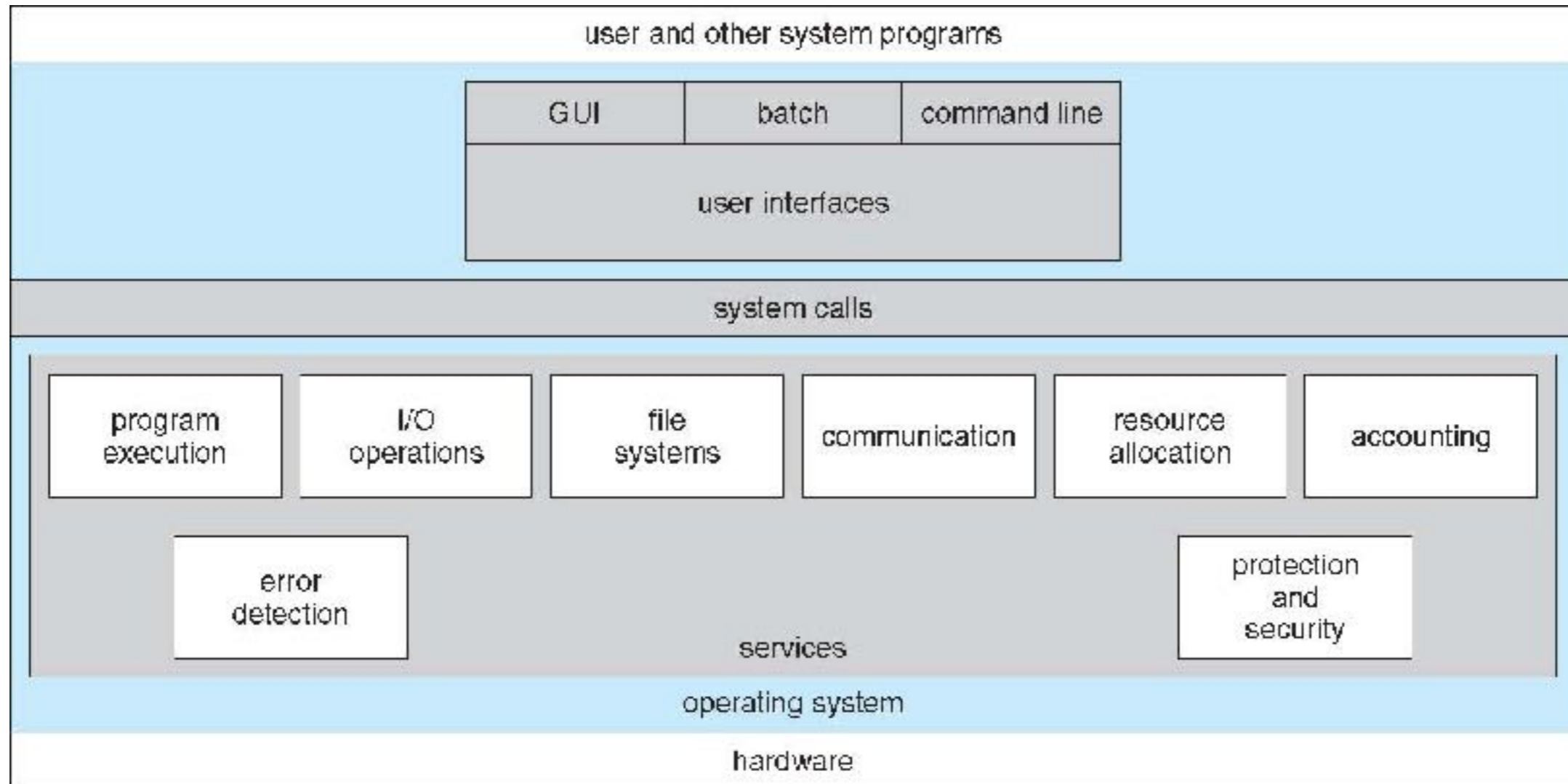
- **Communications** - Les processus peuvent échanger des informations, sur le même ordinateur ou entre ordinateurs sur un réseau
- Les communications peuvent être effectuées via la mémoire partagée ou via le transfert de messages (paquets déplacés par le système d'exploitation)

# Les Services (Utilisateur)



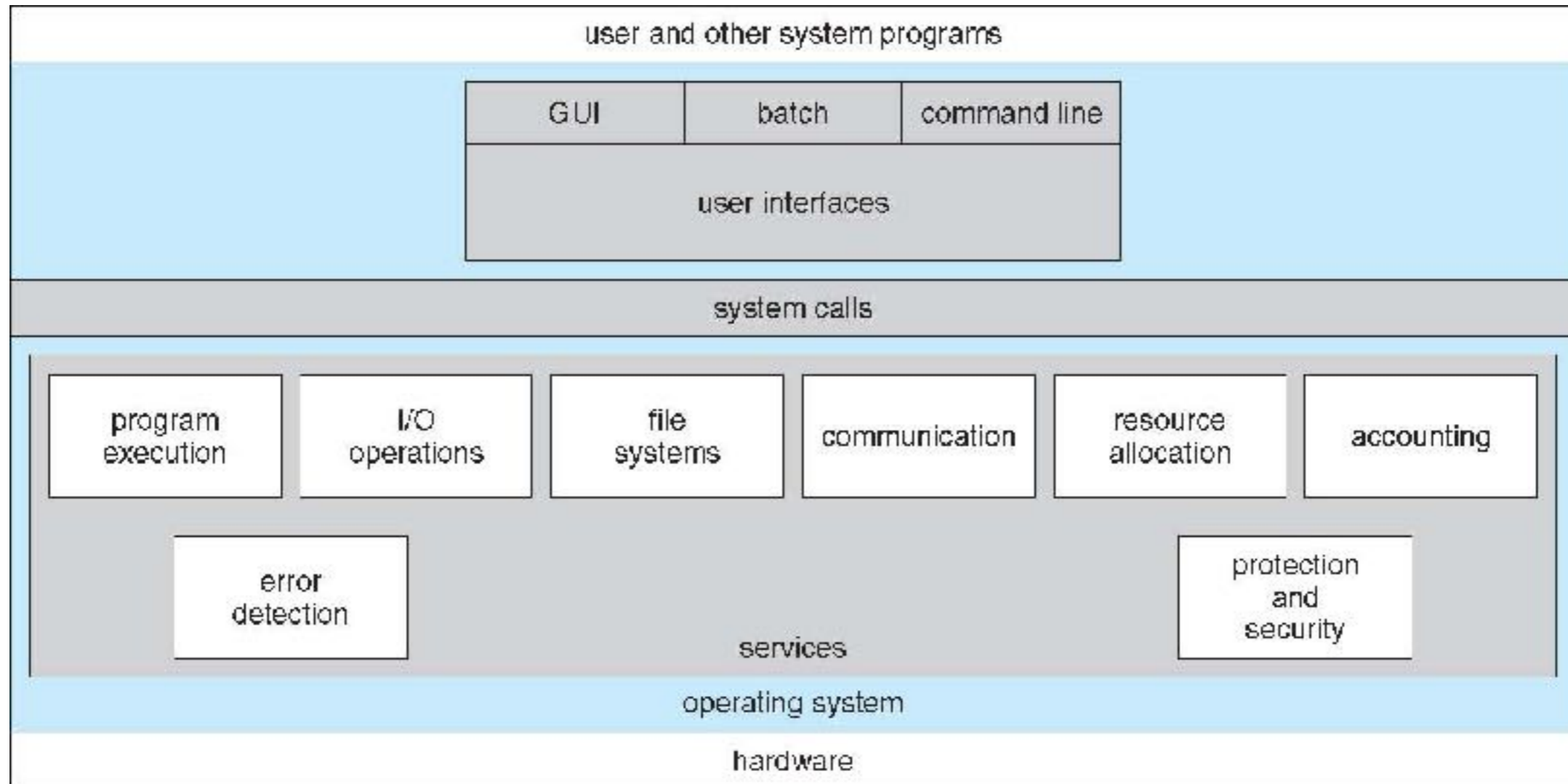
- **Détection d'erreurs** - Le système d'exploitation doit être constamment conscient des erreurs possibles
  - Peut se produire dans le processeur et le matériel de mémoire, dans les périphériques d'E / S, dans le programme utilisateur
  - Les installations de débogage peuvent grandement améliorer les capacités de l'utilisateur et du programmeur à utiliser efficacement le système

# Les Services (Système)



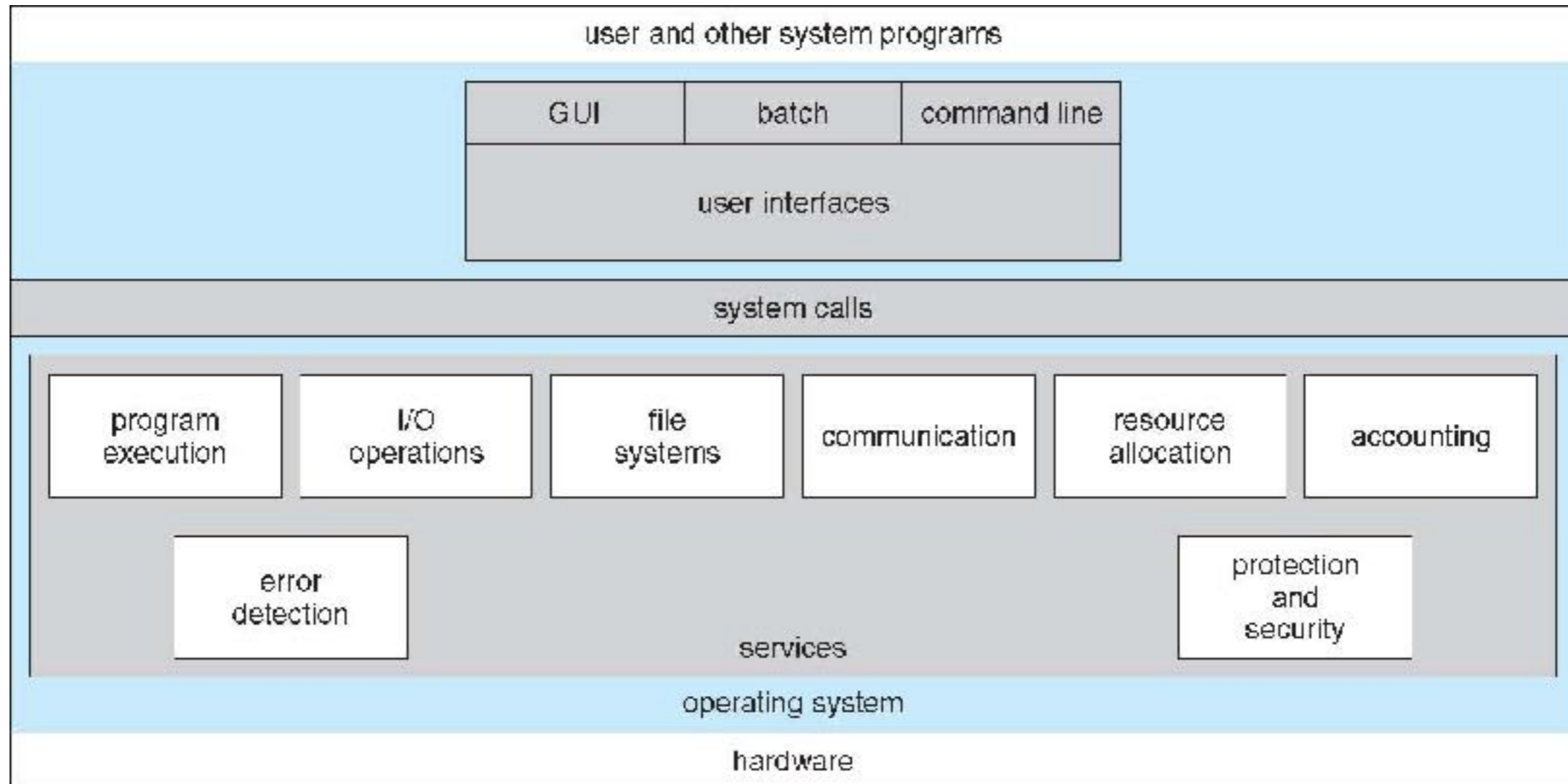
- **Allocation de ressources** - Lorsque plusieurs utilisateurs ou plusieurs jobs s'exécutent simultanément, des ressources doivent être allouées à chacun d'eux
- Nombreux types de ressources et de stratégies - Certains (par exemple cycles d'UC, mémoire principale et stockage de fichiers) peuvent avoir un code d'allocation spécial, d'autres (par exemple, périphériques d'E / S) peuvent avoir une demande générale et un code de libération

# Les Services (Système)



- **Comptabilité** - Pour garder une trace de quels utilisateurs utilisent combien et quels types de ressources informatiques pour la facturation ou les statistiques pour régler les paramètres

# Les Services (Système)



- **Protection et sécurité** - Les propriétaires d'informations stockées dans un système informatique multi-utilisateur ou en réseau peuvent vouloir contrôler l'utilisation de ces informations, les processus concurrents ne doivent pas interférer les uns avec les autres

# Menu

---

- Les services d'un système d'exploitation
- **L'interface d'un système d'exploitation**
- Composants du système d'exploitation
- Démarrage du système informatique

# Les Types d'Interfaces

---

- “Command line” (CLI)
- Graphical user interface (GUI)
- Appel système
- Programme système

# Interface de ligne de commande (CLI)

---

- L'interface de ligne de commande (CLI) ou l'interpréteur de commandes permet une entrée de commande directe
  - Parfois implémenté dans le noyau, parfois par programme système
  - Parfois, plusieurs saveurs mises en œuvre - “shells”
  - Récupère principalement une commande de l'utilisateur et l'exécute
    - ✓ Parfois des commandes intégrées, parfois juste des noms de programmes
      - Si ce dernier, l'ajout de nouvelles fonctionnalités ne nécessite pas de modification shell



# Ex: Bourne Shell Command Interpreter

```

Default
New Info Close Execute Bookmarks
Default Default
PBG-Mac-Pro:~ pbg$ w
15:24 up 56 mins, 2 users, load averages: 1.51 1.53 1.65
USER      TTY      FROM          LOGIN@   IDLE WHAT
pbg       console  -             14:34    50  -
pbg       s000    -             15:05    -  w
PBG-Mac-Pro:~ pbg$ iostat 5
          disk0          disk1          disk10         cpu         load average
      KB/t tps MB/s    KB/t tps MB/s    KB/t tps MB/s  us sy id  1m  5m  15m
      33.75 343 11.30    64.31 14  0.88    39.67  0  0.02  11  5 84  1.51 1.53 1.65
      5.27 320  1.65     0.00  0  0.00     0.00  0  0.00   4  2 94  1.39 1.51 1.65
      4.28 329  1.37     0.00  0  0.00     0.00  0  0.00   5  3 92  1.44 1.51 1.65
^C
PBG-Mac-Pro:~ pbg$ ls
Applications                Music
Applications (Parallels)    Pando Packages
Desktop                      Pictures
Documents                   Public
Downloads                   Sites
Dropbox                     Thumbs.db
Library                     Virtual Machines
Movies                      Volumes
WebEx
config.log
getsmartdata.txt
imp
log
panda-dist
prob.txt
scripts
PBG-Mac-Pro:~ pbg$ pwd
/Users/pbg
PBG-Mac-Pro:~ pbg$ ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1): 56 data bytes
64 bytes from 192.168.1.1: icmp_seq=0 ttl=64 time=2.257 ms
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=1.262 ms
^C
--- 192.168.1.1 ping statistics ---
2 packets transmitted, 2 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 1.262/1.760/2.257/0.498 ms
PBG-Mac-Pro:~ pbg$ █

```

# Graphical User Interface (GUI)

---

- Interface convivial: de métaphore de “desktop”
  - Habituellement souris, clavier et moniteur
  - Les icônes représentent les fichiers, les programmes, les actions, etc.
  - Divers boutons de la souris sur les objets de l'interface provoquent diverses actions (fournir des informations, des options, exécuter la fonction, ouvrir le répertoire (connu sous le nom d'un dossier))
  - Invented a Xerox PARC
  
- De nombreux systèmes incluent maintenant des interfaces CLI et GUI

# GUI: “Touchscreen”

- Les appareils à écran tactile nécessitent de nouvelles interfaces
  - Souris impossible ou non souhaitée
  - Actions et sélection basées sur des gestes
  - Clavier virtuel pour la saisie de texte



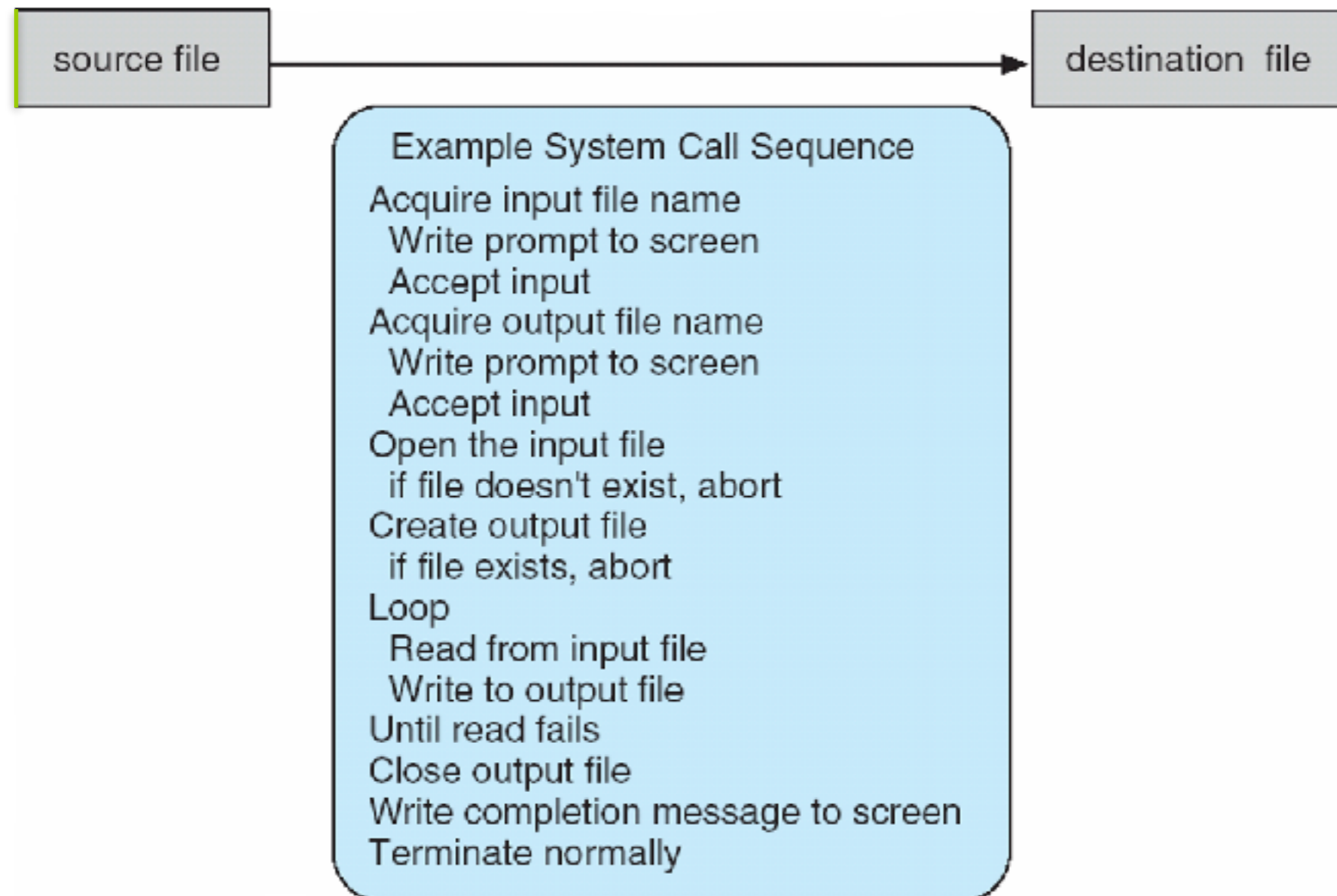
# Interface Appel Système

---

- Interface de programmation pour les services fournis par le système d'exploitation
- Principalement accessible par les programmes via une interface de programme d'application (API) de haut niveau plutôt que par l'utilisation directe des appels système
- Les 3 plus utiliser sont:
  - Win32 API for Windows,
  - **POSIX API for POSIX-based systems (including virtually all versions of UNIX, Linux, and Mac OS X), and**
  - Java API for the Java virtual machine (JVM)

# Exemple d'un Appel Système

- Séquence d'appel système pour copier le contenu d'un fichier dans un autre fichier



# Example of Standard API

## EXAMPLE OF STANDARD API

As an example of a standard API, consider the `read()` function that is available in UNIX and Linux systems. The API for this function is obtained from the `man` page by invoking the command

```
man read
```

on the command line. A description of this API appears below:

```
#include <unistd.h>

ssize_t read(int fd, void *buf, size_t count)
```

return value	function name	parameters

A program that uses the `read()` function must include the `unistd.h` header file, as this file defines the `ssize_t` and `size_t` data types (among other things). The parameters passed to `read()` are as follows:

- `int fd`—the file descriptor to be read
- `void *buf`—a buffer where the data will be read into
- `size_t count`—the maximum number of bytes to be read into the buffer

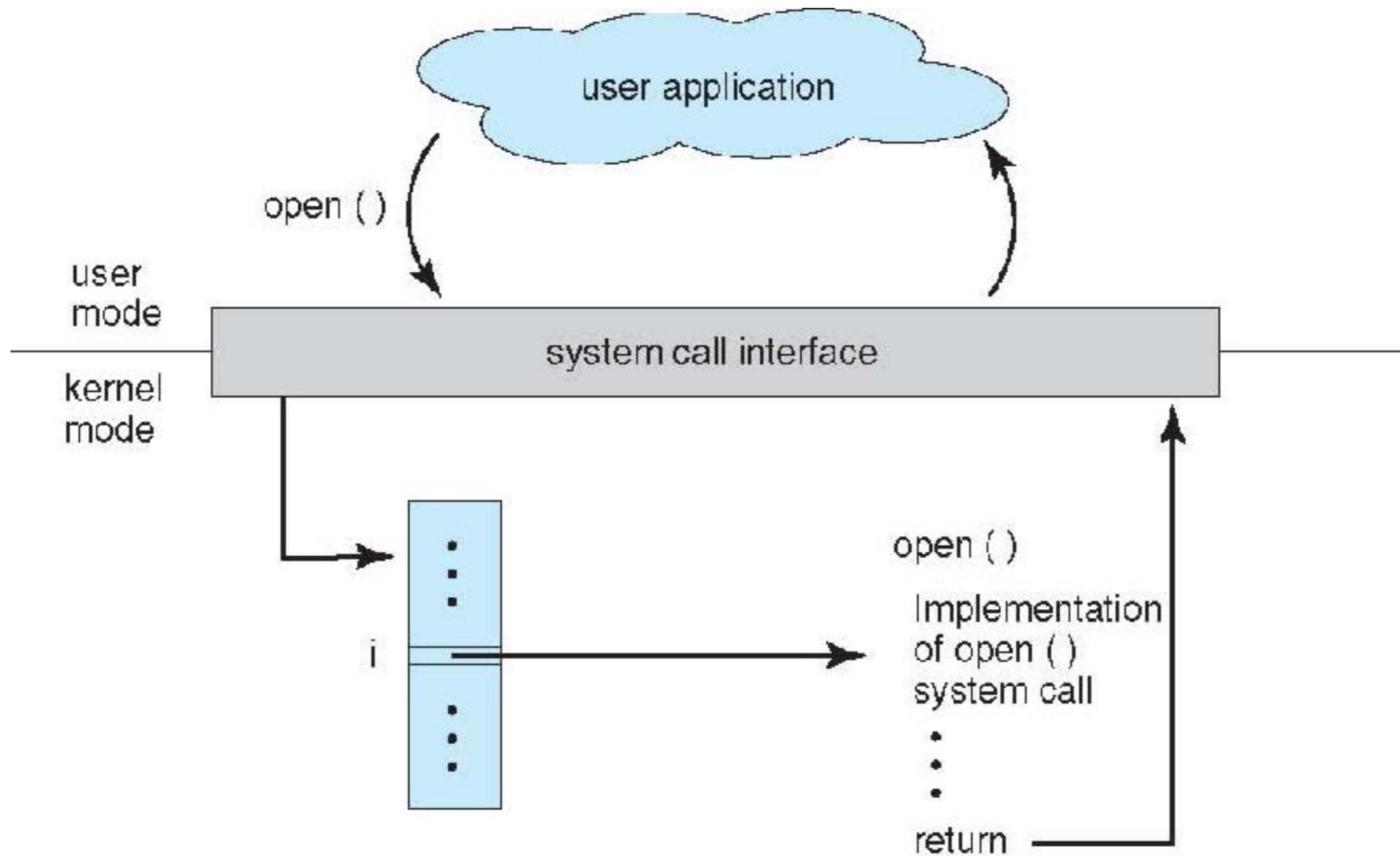
On a successful read, the number of bytes read is returned. A return value of 0 indicates end of file. If an error occurs, `read()` returns `-1`.

# Interface Appel Système

---

- Généralement, un numéro associé à chaque appel système
  - L'interface d'appel système maintient une table indexée selon ces chiffres
- L'interface d'appel système appelle l'appel système prévu dans le noyau du système d'exploitation et renvoie l'état de l'appel système et toutes les valeurs de retour
- L'appelant n'a pas besoin de savoir comment l'appel système est implémenté
  - La plupart des détails de l'interface du système d'exploitation cachés du programmeur par l'API

# Interface Appel Système



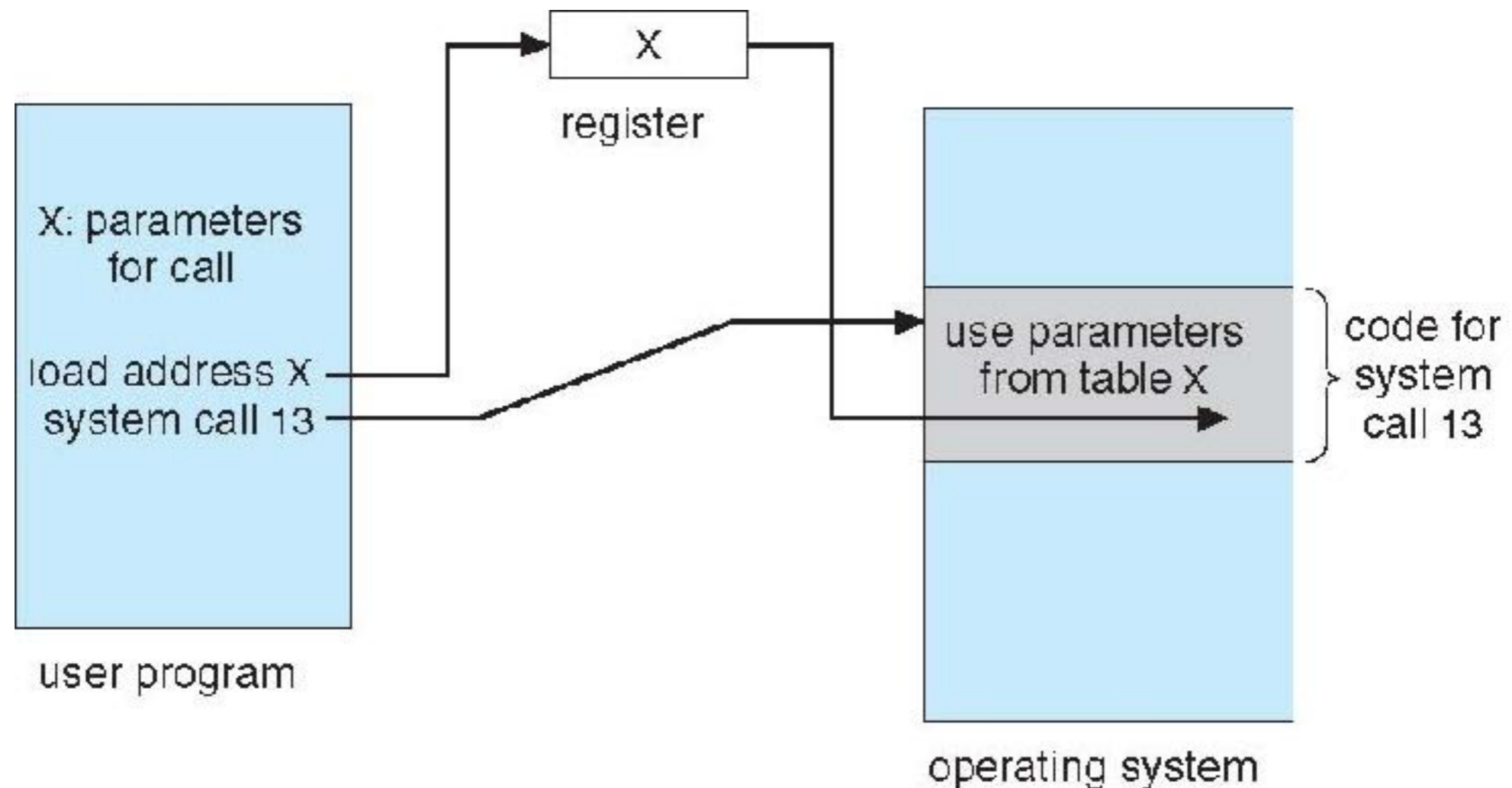


# Appel Système: Transmettre les Paramètre

---

- Souvent, plus d'informations sont nécessaires que simplement l'identité de l'appel système souhaité
  - Le type exact et la quantité d'informations varient en fonction du système d'exploitation et de l'appel
- Trois méthodes générales utilisées pour transmettre des paramètres au système d'exploitation
  - Le plus simple: passer les paramètres dans les registres
    - ✓ Dans certains cas, il peut y avoir plus de paramètres que de registres
  - Paramètres stockés dans un bloc, ou une table, dans la mémoire, et l'adresse du bloc passé en paramètre dans un registre
    - ✓ Cette l'approche prise par Linux et Solaris
  - Paramètres placés ou poussés sur la pile par le programme et supprimés de la pile par le système d'exploitation

# Appel Système: Transmettre les Paramètre



# Types d'Appel Système

---

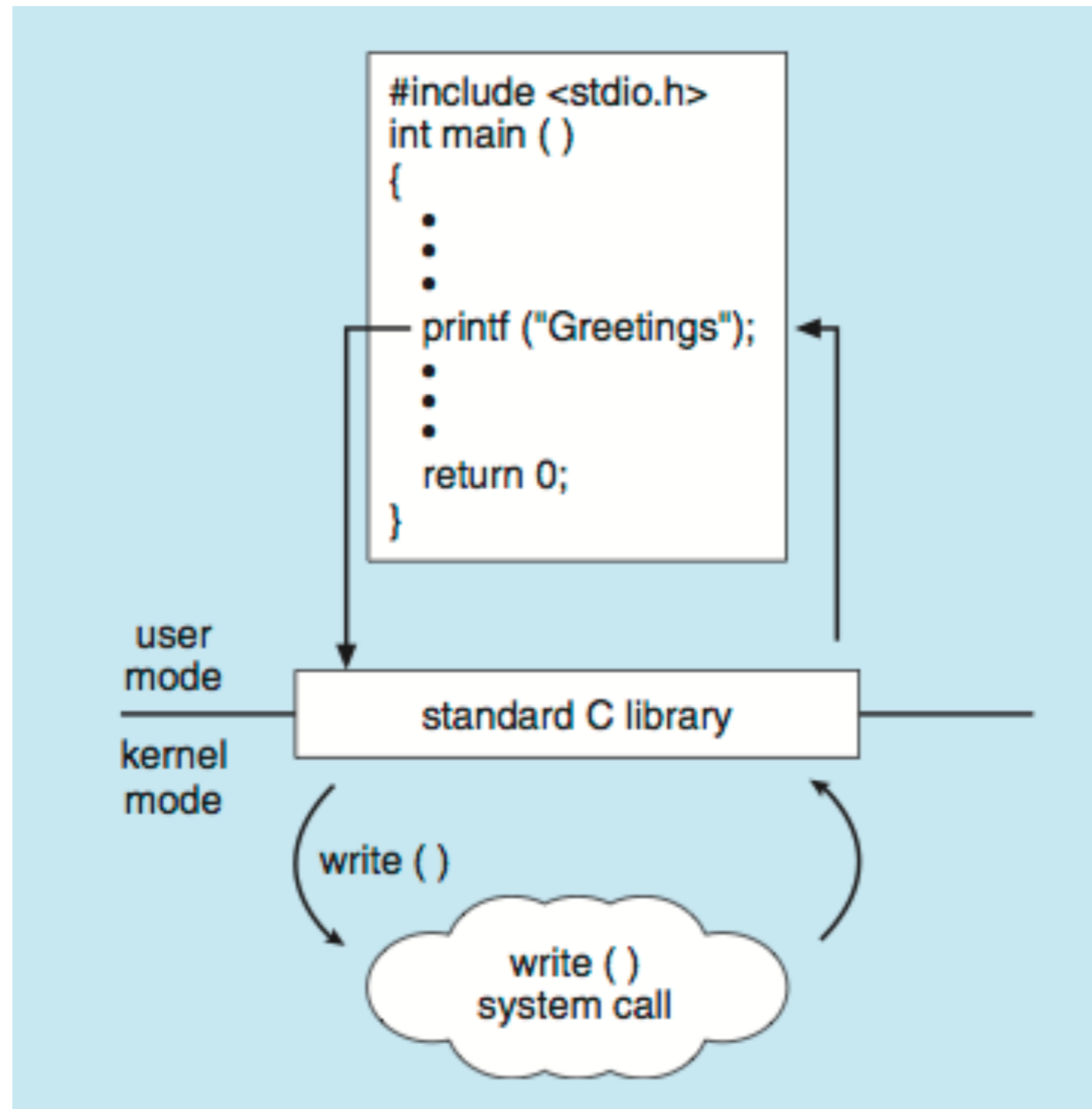
- Contrôle de processus
- Gestion des fichiers
- Gestion des appareils
- Maintenance de l'information
- Communication
- Protection

# Types d'Appel Système: Contrôle de processus

---

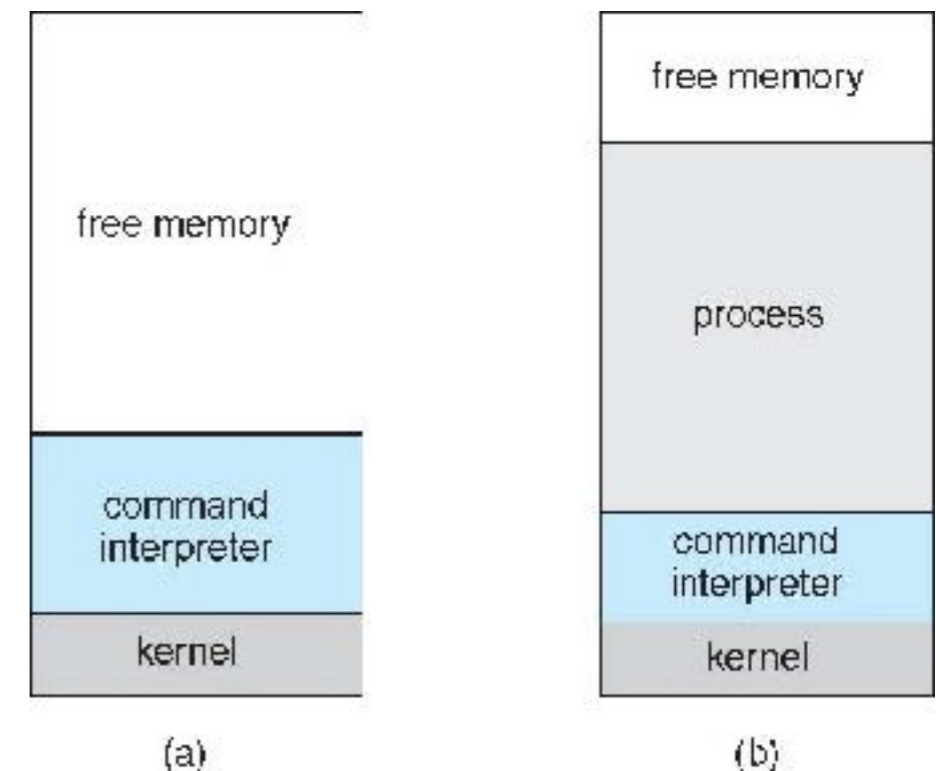
- fin, avorter
- charger, exécuter
- creation/termination de processus
- obtenir/définir des attributs de processus
- attendez pour un certain temps
- attendez/signalez un événement
- gestion de la memoire
- “memory dump” dans le cas d’erreur
- débogage
- Verrous pour gérer l'accès aux données partagées entre les processus

# Ex1: “Hello World”



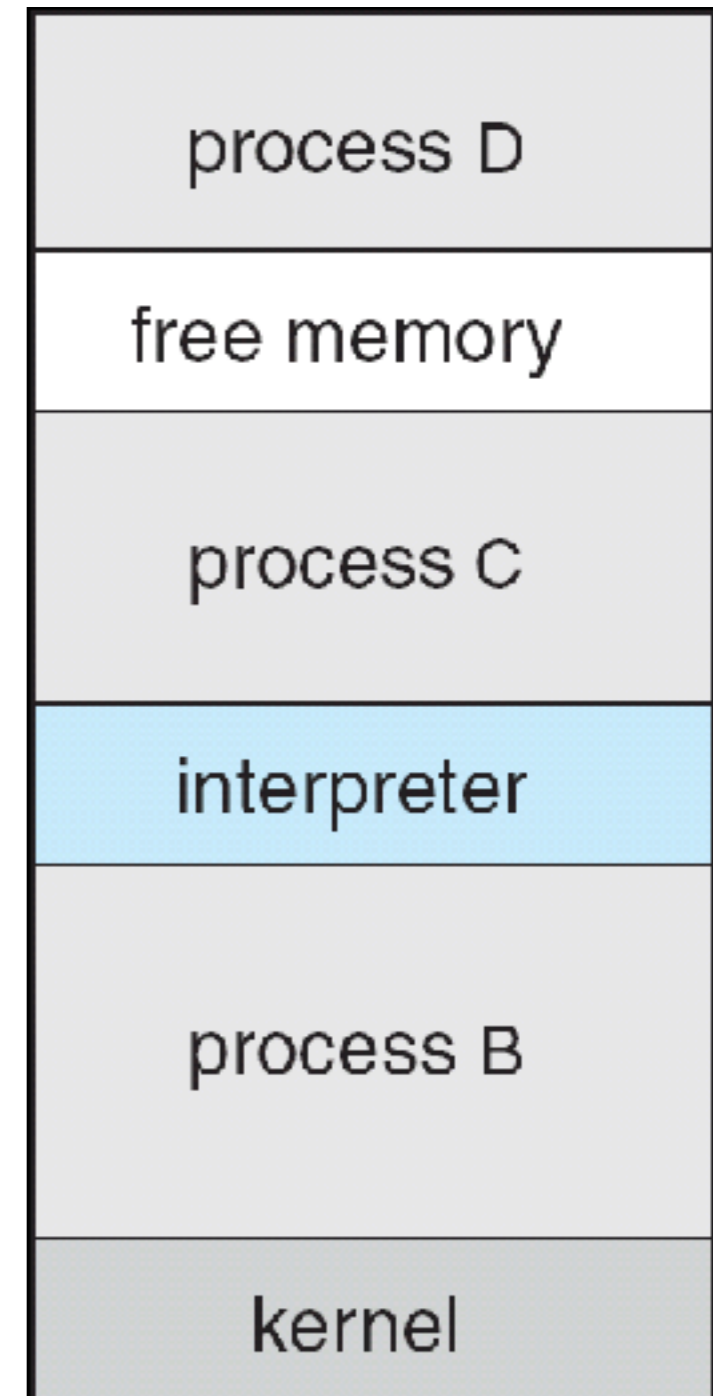
# Example: MS-DOS

- Single-Tâches
- Shell invoqué lors du démarrage du système
- Méthode simple pour exécuter le programme
  - Aucun processus créé
- Espace mémoire unique
- Charge le programme en mémoire, écrasant tout sauf le noyau
- Sortie du programme -> shell rechargée



# Example: FreeBSD

- Unix variant
- Multitasking
- User login -> invoke use's choice of shell
- Shell executes `fork()` system call to create process
  - Executes `exec()` to load program into process
  - Shell waits for process to terminate or continues with user commands
- Process exits with code of 0 – no error or  $> 0$  – error code



# Types d'Appel Système: Gestion de fichiers

---

- créer/supprimer les fichiers
- ouvrir/fermer les fichiers
- lire, écrire, repositionner
- obtenir et définir les attributs de fichier



# Types d'Appel Système: Gestion des Appareils

---

- demande, libération de l'appareil
- lire, écrire, repositionner
- obtenir/définir les attributs de l'appareil
- logiquement connecter ou détacher des périphériques

# Types d'appel système: Maintenance de l'information

- obtenir/définir la temps ou la date
- obtenir/définir les données du système

# Types d'appel système: Communications

---

## ■ Communications

- créer/supprimer les connexions
- envoyer, recevoir des messages si le message transmettant le modèle au nom d'hôte ou au nom du processus
  - ✓ Du client (source) au serveur (récepteur daemon) (e.g. `sshd`)
- Le modèle à mémoire partagée crée et accède aux régions de la mémoire
- transférer les informations d'état
- attacher et détacher des périphériques distants

# Types d'appel système

---

- Protection
  - Contrôler l'accès aux ressources
  - Obtenir et définir des autorisations
  - Autoriser et refuser l'accès des utilisateurs à certaines ressources

# Exemples

	Windows	Unix
<b>Contrôle de processus</b>	CreateProcess() ExitProcess() WaitForSingleObject()	fork() exit() wait()
<b>Gestion des fichiers</b>	CreateFile() ReadFile() WriteFile() CloseHandle()	open() read() write() close()
<b>Gestion des appareils</b>	SetConsoleMode() ReadConsole() WriteConsole()	ioctl() read() write()
<b>Maintenance de l'Information</b>	GetCurrentProcessID() SetTimer() Sleep()	getpid() alarm() sleep()
<b>Communications</b>	CreatePipe() CreateFileMapping() MapViewOfFile()	pipe() shmget() mmap()
<b>Protections</b>	SetFileSecurity() ...	chmod() umask() chown()



# Programmes Système

---

- Les programmes système (utilitaires) fournissent un environnement pratique pour le développement et l'exécution du programme. Ils peuvent être divisés en:
  - Manipulation de fichier
  - Informations d'état parfois stockées dans une modification de fichier
  - Support de la langue de programmation
  - Program loading and execution
  - Communications
  - Service d'arrière-plan
  - Programmes applications
- La vue de la plupart des utilisateurs du système d'exploitation est définie par les programmes système et les programmes d'application, et non par les appels système réels

# Programmes Système

---

- Gestion des fichiers:
  - Créer, supprimer, copier, renommer, imprimer, vider, répertorier et manipuler généralement des fichiers et des répertoires (`touch`, `mkdir`, `cp`, ...)
- Informations d'état
  - date, heure, quantité de mémoire disponible, espace disque, nombre d'utilisateurs (`date`)
  - informations sur les performances, la journalisation et le débogage
- Modifications des fichiers
  - éditeur de texte (`vim`, `emacs`, `nano`)
  - Commandes spéciales pour rechercher le contenu des fichiers ou effectuer des transformations du texte (`grep`, `sed`)

# Programmes Système

---

- **Prise en charge de la programmation** - Compilateurs, assembleurs, débogueurs et interprètes parfois fournis (`gcc`, `gdb`, `python`)
- **Chargement et exécution de programme** - Chargeurs absolus, chargeurs relocalisables, éditeurs de liaison et chargeurs de superposition, systèmes de débogage pour le langage de haut niveau et le langage machine
- **Communications** - Fournit le mécanisme de création de connexions virtuelles entre les processus, les utilisateurs et les systèmes informatiques



# Programmes Système

---

- Services “background”
  - Lancement au démarrage
    - ✓ Certains pour le démarrage du système, puis se terminent
    - ✓ Certains du démarrage du système à l'arrêt
  - Fournir des fonctionnalités telles que la vérification du disque, la planification des processus, la consignation des erreurs, l'impression
  - Exécuter dans le contexte de l'utilisateur et non dans le contexte du noyau
  - Connu sous le nom de services, sous-systèmes, démons
  
- Programmes applications
  - Pas une partie du système d'exploitation

# Menu

---

- Les services d'un système d'exploitation
- L'interface d'un système d'exploitation
- **Composants du système d'exploitation**
- Démarrage du système informatique

# Composants du Système d'Exploitation

---

- Implementation
- Structure
- Déboggage
- Génération

# Conception du système d'exploitation

---

- Pas juste **une façon** de le faire mais il y a des stratégies qui fonctionnent bien
- La structure interne des différents systèmes d'exploitation peut varier considérablement
- Commencez par définir les objectifs et les spécifications
- Affecté par le choix du matériel, type de système
- Buts de l'utilisateur et objectifs du système
  - Buts de l'utilisateur - le système d'exploitation devrait être pratique à utiliser, facile à apprendre, fiable, sûr et rapide
  - Objectifs du système - le système d'exploitation devrait être facile à concevoir, à mettre en œuvre et à maintenir, ainsi que flexible, fiable, sans erreur et efficace

# Conception du système d'exploitation

---

- Principe important à séparer:
  - Politique: Que va-t-on faire?
  - Mécanisme: Comment va-t-on faire?
- Les mécanismes déterminent comment faire quelque chose, les politiques décident de ce qui sera fait
  - La séparation de la politique du mécanisme est un principe très important, elle permet une flexibilité maximale si les décisions politiques doivent être changées plus tard.
- Spécifier et concevoir OS est une tâche très créative de génie logiciel

# La Mise en Oeuvre

---

- Beaucoup de variation
- Normalement un mélange de langage de programmation
  - Niveaux les plus bas dans “assembly”
  - Corps principal en C
  - Programmes système en C, C ++, langages de script comme PERL, Python, scripts shell
- Plus de langage de haut niveau plus facile à mettre en communication avec d'autres matériels
  - Mais moins efficace
- L'émulation peut permettre à un système d'exploitation de s'exécuter sur du matériel non natif

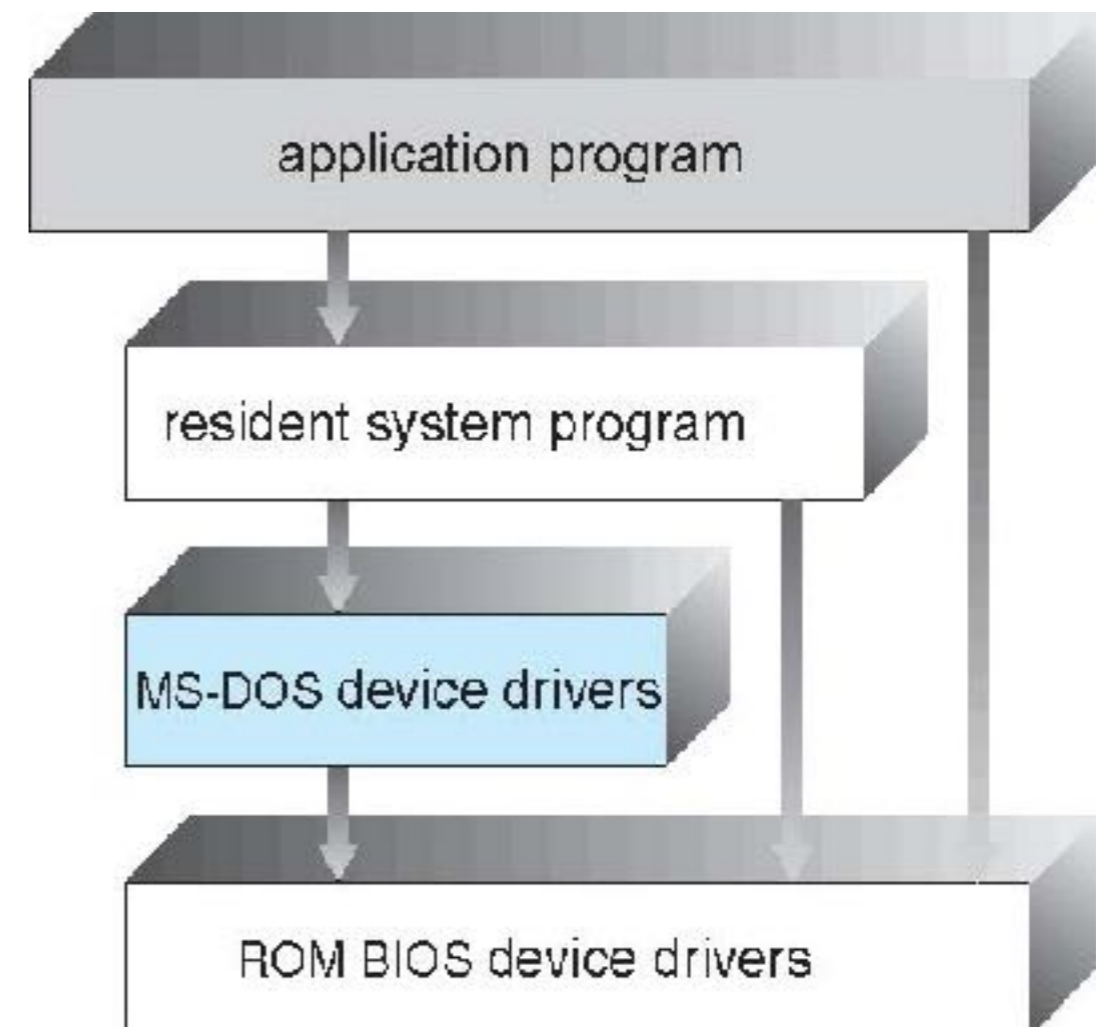
# Structure du Système d'Exploitation

---

- Simple (MS-DOS)
- En couches
- “Microkernels” (Mach)
- Modules
- Systèmes Hybrids

# Structure Simple

- MS-DOS – écrit pour fournir le plus de fonctionnalités dans le moins d'espace
  - Non divisé en modules
  - Bien que MS-DOS ait une certaine structure, ses interfaces et ses niveaux de fonctionnalité ne sont pas bien séparés





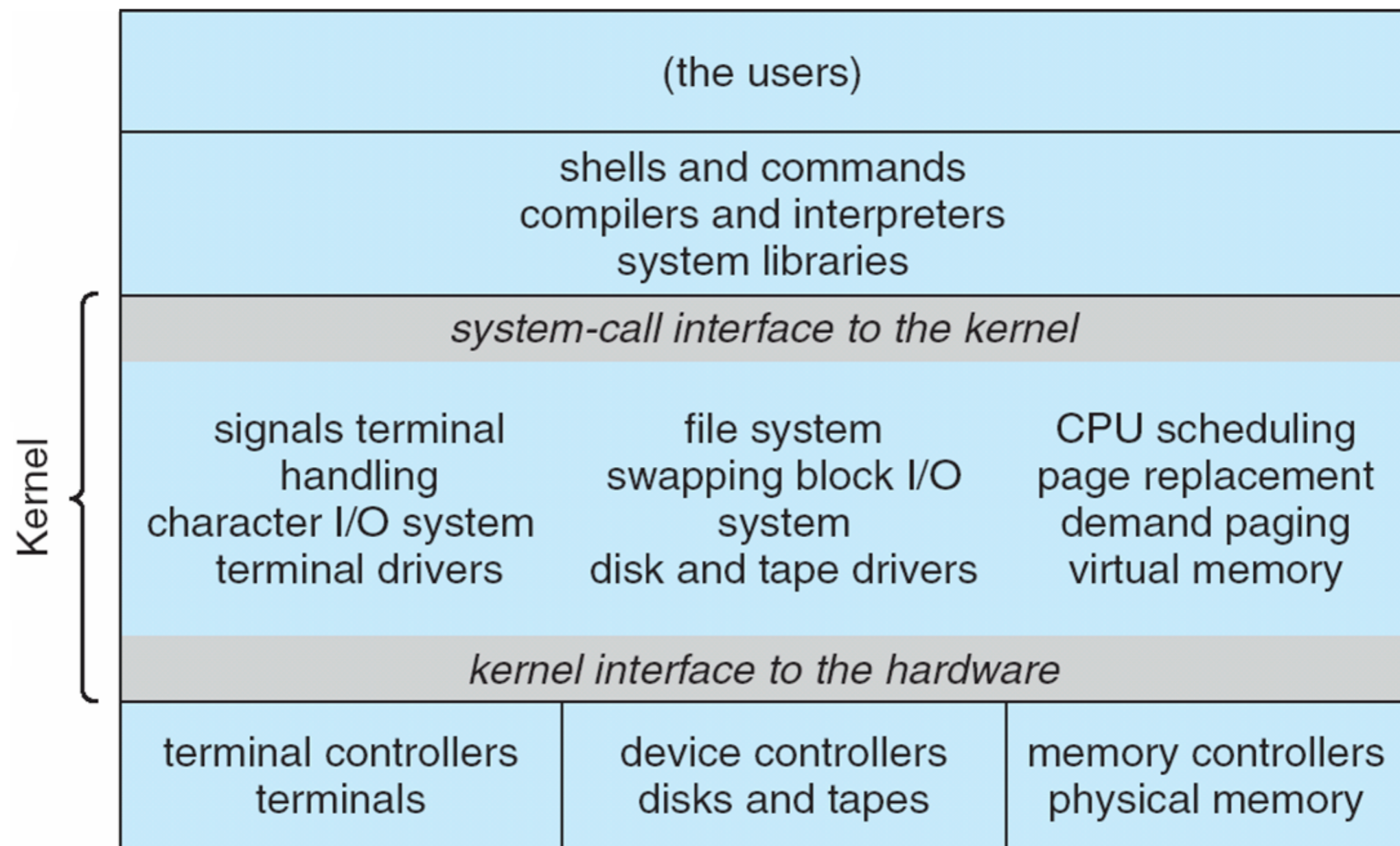
# UNIX

---

- UNIX – Limité par la fonctionnalité matérielle, le système d'exploitation UNIX d'origine avait une structure limitée. Le système d'exploitation UNIX se compose de deux parties séparables:
  - Programmes systèmes
  - Le noyau
    - ✓ Se compose de tout ce qui est en dessous de l'interface d'appel système et au-dessus du matériel physique
    - ✓ Fournit le système de fichiers, la planification de l'UC, la gestion de la mémoire et d'autres fonctions du système d'exploitation; un grand nombre de fonctions pour un niveau
    - ✓ Très peu de frais généraux dans l'interface d'appel système ou de communication au sein du noyau, conduisant à un avantage de performance distinct

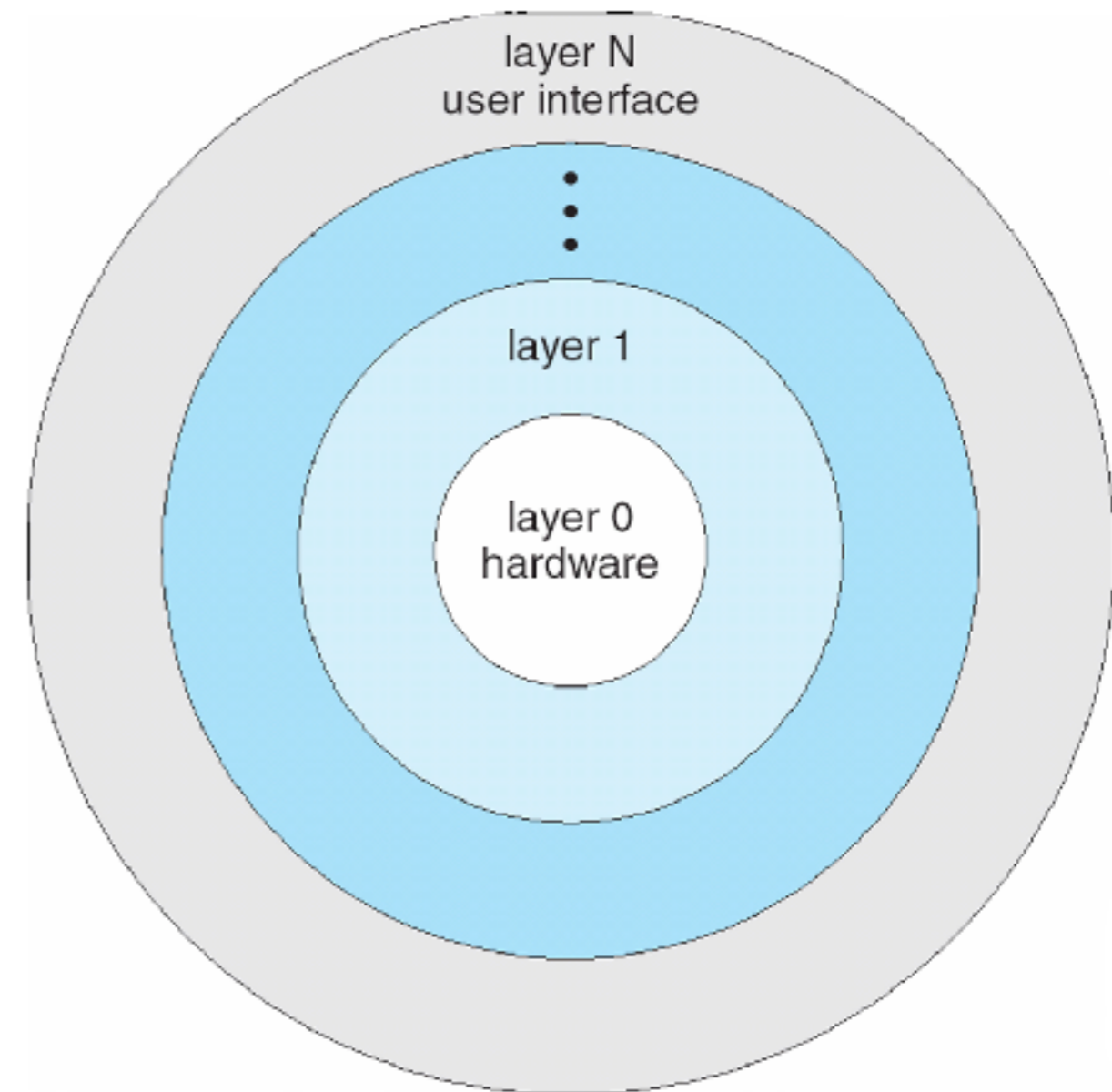
# Structure du Système UNIX Traditionnelles

Au-delà simple mais pas entièrement en couches



# Approche en couches

- Le système d'exploitation est divisé en plusieurs couches (niveaux), chacune étant construite au-dessus des couches inférieures. La couche inférieure (couche 0), est le matériel; le plus haut (couche N) est l'interface utilisateur.
- Grâce à la modularité, les couches sont sélectionnées de manière à ce que chacune utilise des fonctions (opérations) et des services de couches de niveau inférieur seulement.

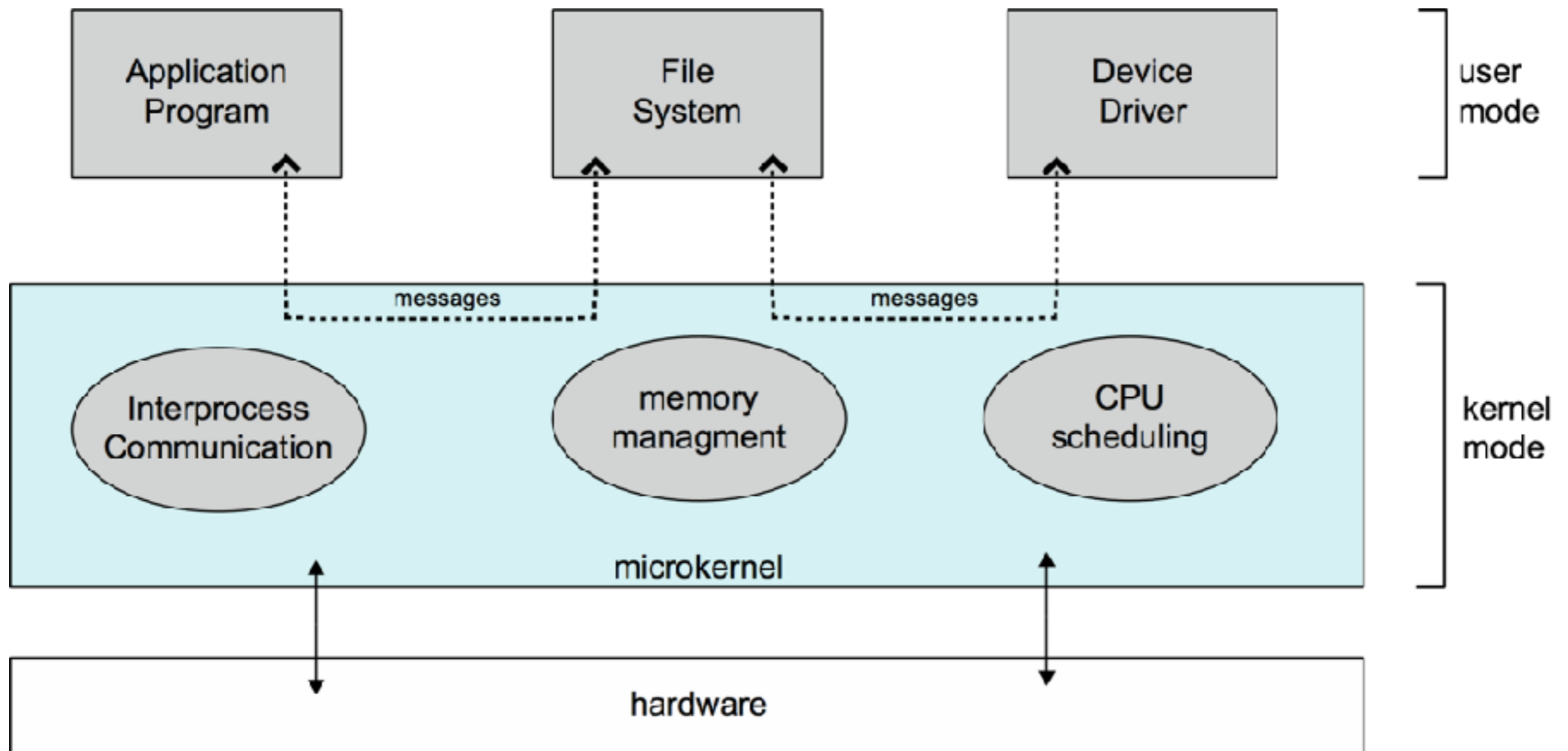


# Structure du système Microkernel

---

- Se déplace autant du noyau vers l'espace utilisateur
- Ex: Mach
  - Mac OS X kernel (Darwin) basé sur Mach
- La communication a lieu entre les modules utilisateurs en utilisant le passage de message
- Avantages:
  - Plus facile d'étendre un micro-noyau
  - Plus facile de porter le système d'exploitation à de nouvelles architectures
  - Plus fiable (moins de code s'exécute en mode noyau)
  - Plus sécurisé
- Désavantage:
  - Augmentation des performances de l'espace utilisateur pour la communication de l'espace noyau

# Structure de “Microkernel”

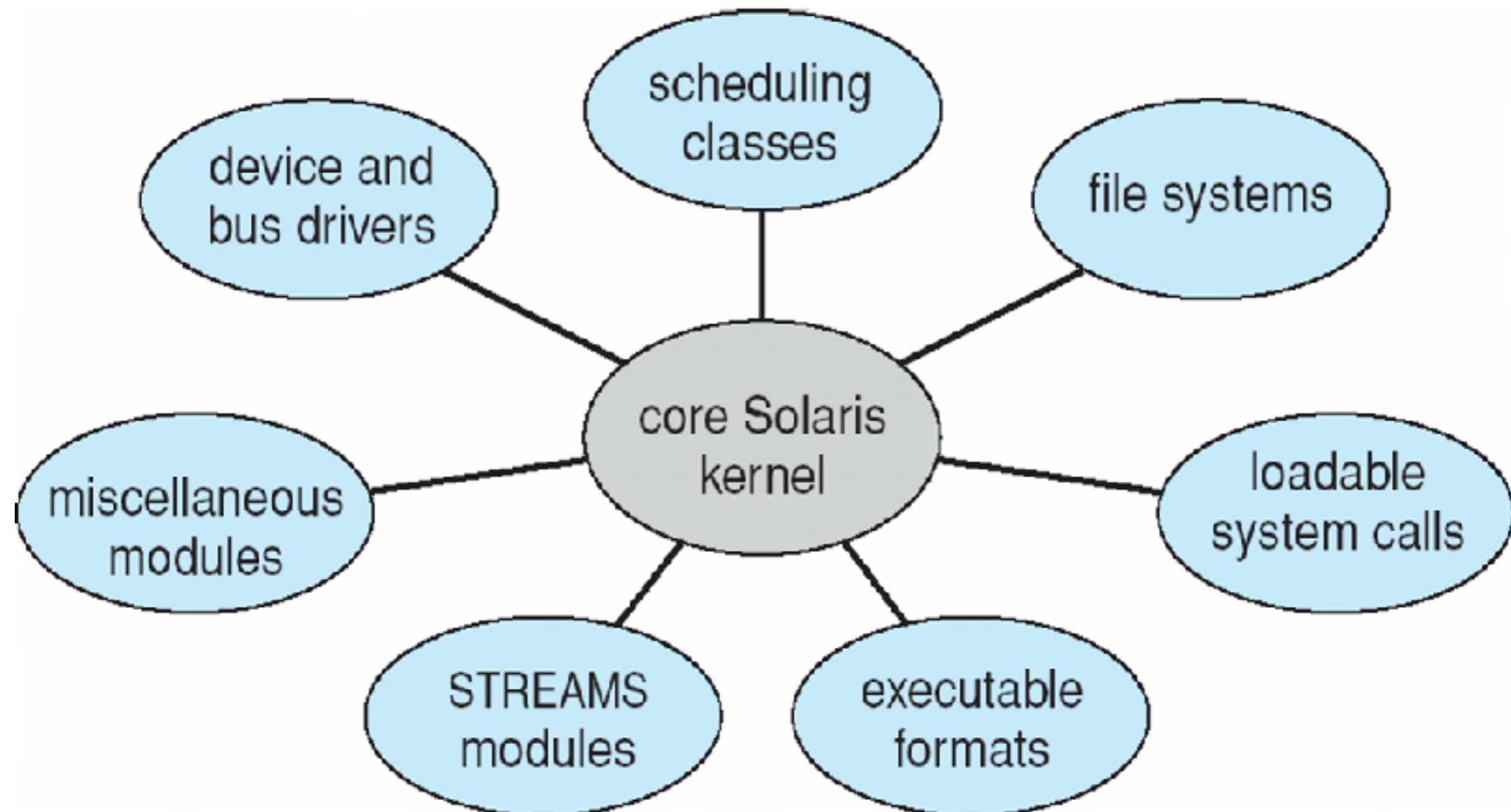


# Modules

---

- La plupart des systèmes d'exploitation modernes implémentent des modules de noyau chargeables
  - Utilise une approche orientée objet
  - Chaque composant de base est séparé
  - Chacun parle aux autres sur des interfaces connues
  - Chacun est chargeable au besoin dans le noyau
  
- Dans l'ensemble, similaire à des couches, mais avec plus de flexibilité
  - Linux, Solaris, etc.

# Solaris Modular Approach



# Systemes hybrides

---

- La plupart des systèmes d'exploitation modernes ne sont pas un modèle pur
  - Hybride combine plusieurs approches pour répondre aux besoins de performance, de sécurité et d'utilisabilité
  - Noyaux Linux et Solaris dans l'espace d'adresse unique du noyau, donc monolithique, plus modulaire pour le chargement dynamique des fonctionnalités
  - Windows principalement monolithique, plus microkernel pour différentes personnalités de sous-système



# Débogage du système d'exploitation

---

- Les SE génèrent des fichiers journaux contenant des informations d'erreur
- Échec d'une application peut générer la mémoire de capture du fichier "core dump" du processus

## Kernighan's Law

“Le débogage est deux fois plus difficile que l'écriture du code en premier lieu. Par conséquent, si vous écrivez le code aussi intelligemment que possible, vous n'êtes par définition pas assez intelligent pour le déboguer.”

- Operating system failure can generate crash dump file containing kernel memory

# BSOD

A problem has been detected and windows has been shut down to prevent damage to your computer.

The problem seems to be caused by the following file: SPCMDCON.SYS

PAGE\_FAULT\_IN\_NONPAGED\_AREA

If this is the first time you've seen this Stop error screen, restart your computer. If this screen appears again, follow these steps:

check to make sure any new hardware or software is properly installed. If this is a new installation, ask your hardware or software manufacturer for any windows updates you might need.

If problems continue, disable or remove any newly installed hardware or software. Disable BIOS memory options such as caching or shadowing. If you need to use safe Mode to remove or disable components, restart your computer, press F8 to select Advanced Startup options, and then select Safe Mode.

Technical information:

\*\*\* STOP: 0x00000050 (0xFD3094C2,0x00000001,0xFBFE7617,0x00000000)

\*\*\* SPCMDCON.SYS - Address FBFE7617 base at FBFE5000, DateStamp 3d6dd67c

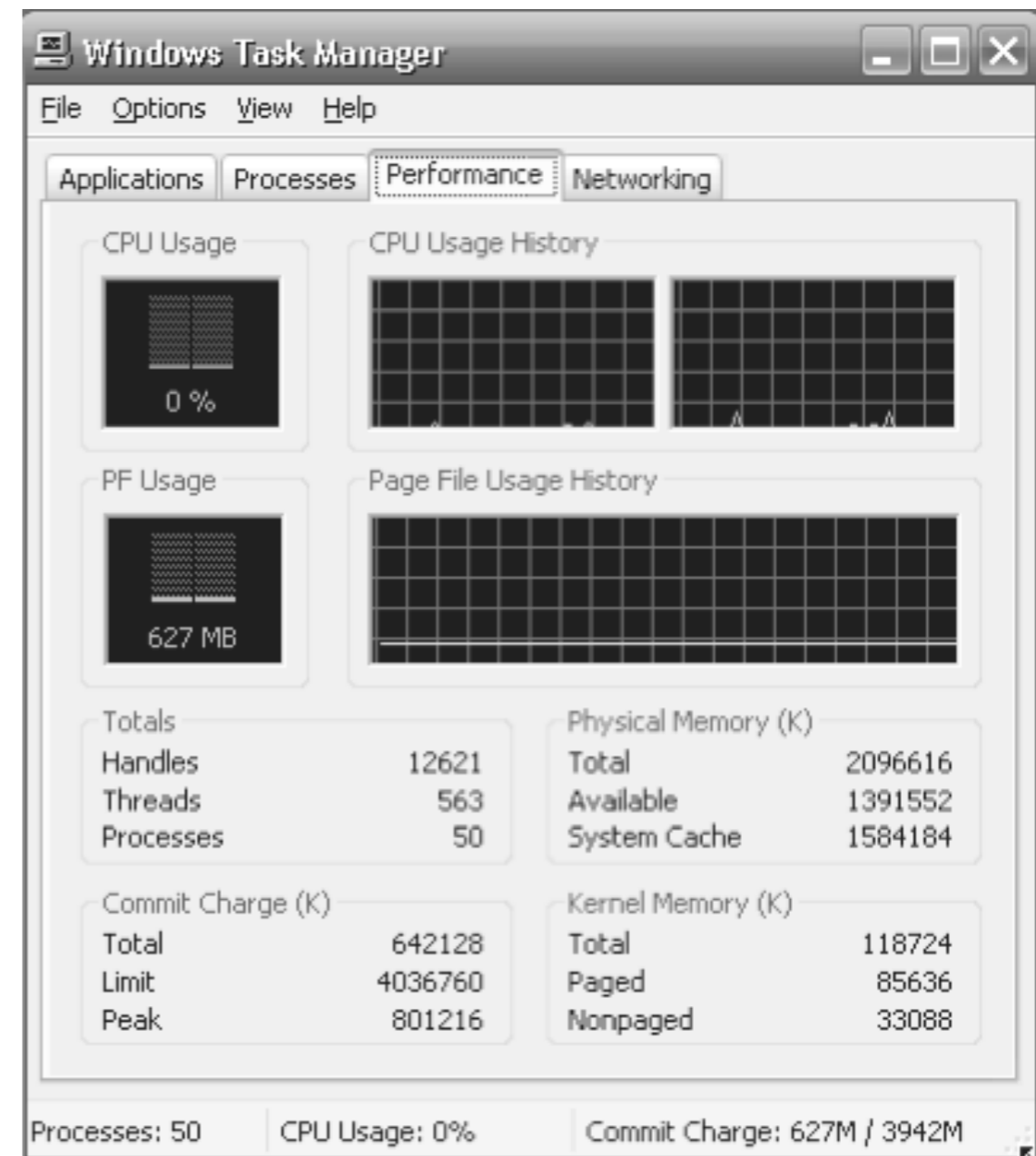
# Utilitaires pour la surveillance du système

```

1  |||||  9.9%]  Tasks: 495, 1831 thr: 1 running
2  |  0.0%]  Load average: 1.63 1.60 1.60
3  |||||  5.3%]  Uptime: 1 day, 16:22:34
4  |  0.0%]
5  |||||  6.0%]
6  |  0.0%]
7  |||  4.0%]
8  ||  0.7%]
Mem ||||| 0.74G/16.0G]
Swp ||||| 110M/1.00G]

PID USER      PRI  NI  VIRT   RES   S  CPU% MEM%  TIME+  Command
49120 lpaul1    17   0 4255M  8496 ?  2.0  0.1  0:30.44 /usr/sbin/screencapture -idcc -tong
48323 lpaul1    24   0 6511M 1299M ?  2.1  7.9  0:50.52 com.docker.hyperkit -A -u -F /Users/lpaul1/L
696 lpaul1    24   0 4732M  298M ?  1.7  1.8  4:25.27 /Applications/Utilities/Terminal.app/Conten
47394 lpaul1    24   0 5541M  462M ?  1.3  2.8  1:27.81 /Applications/Utilities/Google Chrome.app/Conten/Ver
36577 lpaul1    17   0 4831M  119M ?  0.3  0.7  6:20.95 /Applications/Spotify.app/Contents/MacOS/Spo
49123 lpaul1    24   0 4196M  4332 R  0.3  0.9  0:00.96 http
616 lpaul1    17   0 4247M 18648 ?  0.2  0.1  2:27.78 /Applications/Utilities/Adobe Sync/CoreSync/
47255 lpaul1    17   0 5279M  294M ?  0.1  1.8  0:56.55 /Applications/Google Chrome.app/Contents/Mac
410 lpaul1    17   0 4313M 18492 ?  0.1  0.1  1:26.31 /Applications/VMware Fusion.app/Contents/Lib
36587 lpaul1    17   0 5729M  355M ?  0.1  2.2  7:02.26 /Applications/Spotify.app/Contents/Framework
587 lpaul1    24   0 694M  9276 ?  0.1  0.1  0:17.61 /Applications/Utilities/Adobe Application Ma
48318 lpaul1    24   0 4274M 15980 ?  0.1  0.1  0:01.17 com.docker.vpnkit --ethernet id:3 --port id:
47287 lpaul1    17   0 5208M  137M ?  0.1  0.8  0:07.76 /Applications/Google Chrome.app/Contents/Ver
834 lpaul1    17   0 4277M 24552 ?  0.0  0.1  0:20.79 /System/Library/CoreServices/SafariSupport.b
16410 lpaul1    17   0 4381M 15776 ?  0.0  0.1  0:03.05 /Applications/Utilities/Adobe Sync/CoreSync/
16406 lpaul1    17   0 4276M 12980 ?  0.0  0.1  0:12.39 /System/Library/PrivateFrameworks/ViewBridge
34364 lpaul1    17   0 4307M 16120 ?  0.0  0.1  0:01.80 /Applications/Utilities/Adobe Sync/CoreSync/
344 lpaul1    17   0 4803M 89916 ?  0.0  0.5  1:00.10 /System/Library/CoreServices/SystemUIServe.
48358 lpaul1    24   0 4277M 52468 ?  0.0  0.3  0:15.38 /Users/lpaul1/.rbenv/versions/2.2.2/bin/ruby
831 lpaul1    17   0 4248M 21344 ?  0.0  0.1  0:20.80 /usr/libexec/SafariCloudHistoryPushAgent
614 lpaul1    24   0 4389M 40690 ?  0.0  0.2  1:41.44 /Applications/Utilities/Adobe Sync/CoreSync/
635 lpaul1    24   0 4898M 54648 ?  0.0  0.3  0:04.86 /Applications/Utilities/Adobe Creative Cloud
16502 lpaul1    24   0 4356M 17584 ?  0.0  0.1  0:02.31 /System/Library/PrivateFrameworks/CommerceKi
337 lpaul1    17   0 4288M 32488 ?  0.0  0.2  0:18.27 /System/Library/Frameworks/CoreTelephony.fra
399 lpaul1    24   0 4361M 31372 ?  0.0  0.2  0:07.40 /System/Library/PrivateFrameworks/CommerceKi
16405 lpaul1    17   0 5827M  619M ?  0.0  3.8  17:50.39 /Applications/Keynote.app/Contents/MacOS/Key
345 lpaul1    17   0 4436M 55764 ?  0.0  0.3  0:14.40 /System/Library/CoreServices/finder.app/Cont
486 lpaul1    17   0 4359M 18056 ?  0.0  0.1  0:03.37 /System/Library/CoreServices/AppleIDAuthAgen
615 lpaul1    24   0 4923M 61732 ?  0.0  0.4  0:05.68 /Applications/Utilities/Adobe Creative Cloud

```



# Génération de Système d'Exploitation

---

- Les systèmes d'exploitation sont conçus pour fonctionner sur n'importe quelle classe de machines; le système doit être configuré pour chaque site informatique spécifique
- Le programme SYSGEN obtient (automatiquement ou via des requêtes sur le sysadmin) des informations concernant la configuration spécifique du système matériel
  - Utilisé pour créer un noyau compilé spécifique au système ou un système configuré
  - Peut générer un code plus efficace qu'un noyau général
- OS compilé pour la machine spécifique (plus rapide), ou des tables liées aux spécificités identifiées d'une machine (intermédiaire), ou toutes les options possibles sont là, et liées si nécessaire (plus lent mais plus général)
- Est-ce que le système d'exploitation doit être recompilé à chaque changement, ou s'adapte-t-il lorsque des changements surviennent?

# Menu

---

- Les services d'un système d'exploitation
- L'interface d'un système d'exploitation
- Composants du système d'exploitation
- **Démarrage du système informatique**

# System Boot

---

- When power initialized on system, execution starts at a fixed memory location (ex: early Intel: 000FFFF0h)
  - Firmware ROM used to hold initial boot code

Étapes:

1. BIOS: Vérifications du système
2. Boot loader: e.g. GRUB - peut sélectionner un SE
3. Noyau du SE commence

# Sommaire

---

- Les systèmes d'exploitation fournissent des services pour:
  - Les utilisateur
  - Les administrateurs
- Les interface entre le système d'exploitation et les utilisateur:
  - Command line
  - GUI
  - Appels système
  - Programmes système
- Il y a des différents modèles pour la creation d'un nouveau système d'exploitations