

# Principes des systèmes d'exploitation

DIPLÔME UNIVERSITAIRE DE TECHNOLOGIE – INFORMATIQUE

PROGRAMME PÉDAGOGIQUE NATIONAL

SEMESTRE 3 / UE 31 : INFORMATIQUE AVANCÉE

CHAMP DISCIPLINAIRE : ARCHITECTURE MATÉRIELLE – SYSTÈMES D'EXPLOITATION - RÉSEAUX





# Objectifs

---

Comprendre l'architecture d'un système d'exploitation, notamment multitâches

Compétences visées / compétences citées dans le référentiel d'activités et de compétences pour les activités :

- FA2-A : Administration de systèmes, de logiciels et de réseaux
- FA2-B : Conseil et assistance technique à des utilisateurs, clients, services
- FA1-C : Réalisation d'une solution informatique

Prérequis : M2101 (Architecture et Programmation des mécanismes de base d'un système informatique)

# Contenus

---

- Partage des ressources (par exemple, ordonnancement)
- Système de gestion de fichiers
- Hiérarchie de la mémoire (dont mécanismes de pagination, mémoire virtuelle, caches)
- Mise en œuvre des tâches : processus lourds et légers (threads)
- Systèmes d'entrée-sortie
- Introduction à la programmation réseau (mise en œuvre de la bibliothèque sockets)

# Modalités de mise en œuvre

---

Programmation de l'API système en C principalement sous Linux

Prolongements possibles :

- Programmation de scripts évolués
- Mesures de performances
- Résolution de problèmes d'interblocage

Mots clés :

Programmation concurrente ; Mémoire virtuelle ; Entrées/Sorties



# Évaluation

---

- Travail au cours de chaque TP évalué par l'enseignant
  - Certains exercices sont à rendre à la fin du TP
  - Certains TP débutent par un test écrit
- } (Coefficient 1)
- Une Interrogation Écrite (IE) (Coeff 0,75) après 6 séances (11 en tout)
  - Un Devoir Surveillé (DS) (Coeff 0,75) en fin de semestre (février)
  - Coefficient total : 2,5 (12 pour UE31, 30 pour le S3)
-

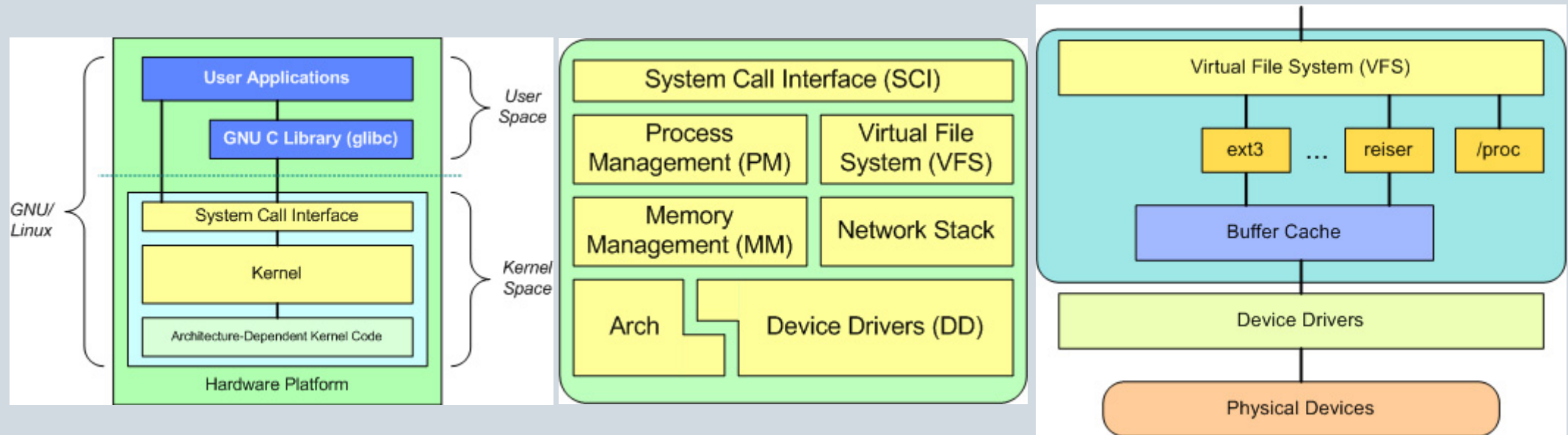
---

# Systemes d'exploitation: introduction, outils et environnement

Compilation et exécution d'un programme

# Rôle du système d'exploitation

Interface entre les ressources et les applications pour permettre notamment de les partager





# Exécution d'un programme

```
syska@dmz-linserv:~$ pstree
```

```
init—NetworkManager—{NetworkManager}  
    |  
    |—accounts-daemon—{accounts-daemon}  
    |  
    |—acpid
```

```
...
```

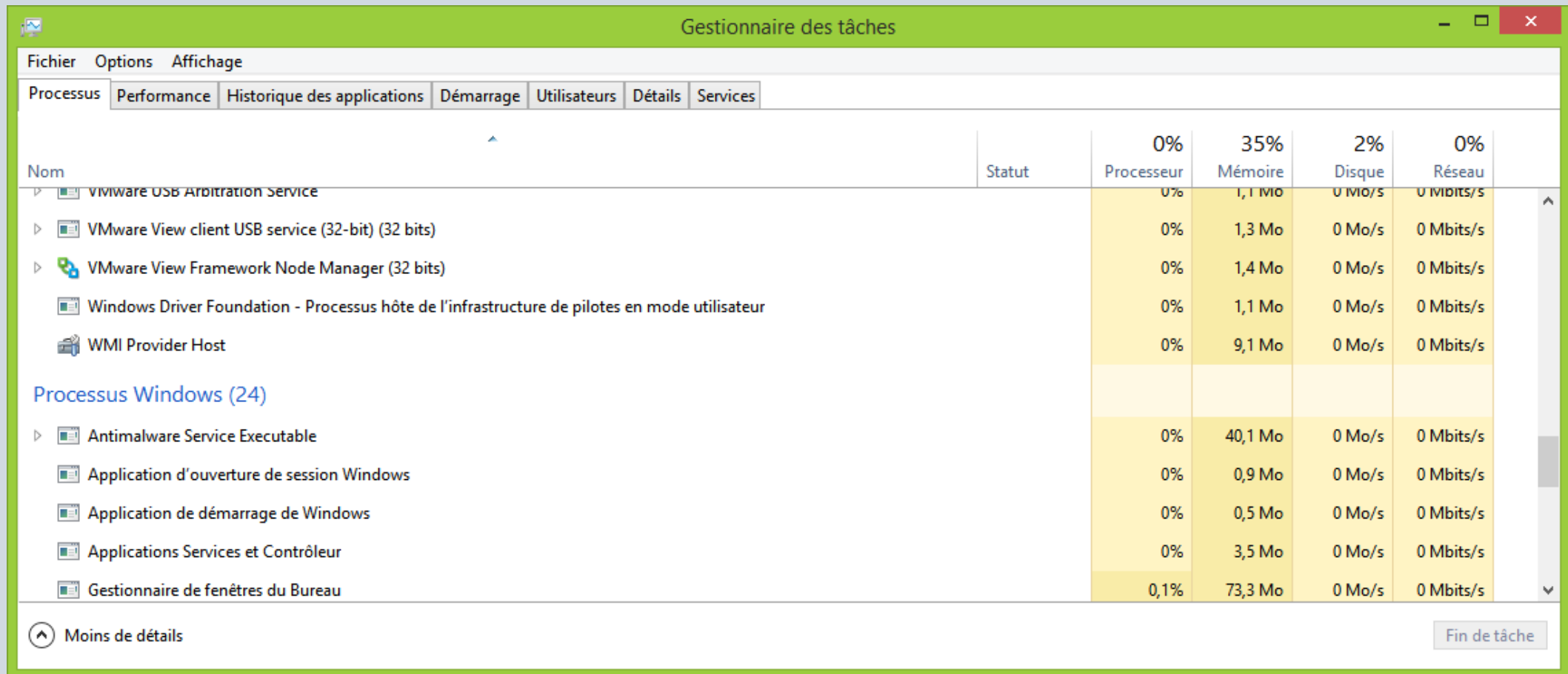
```
    |—sshd—sshd—sshd—usershell—pstree  
    |—udev—2*[udev]  
    |—upower—2*[{upower}]
```



# Exécution d'un programme

```
systemd—┬─ModemManager——2*[{ModemManager}]
          │
          └─NetworkManager—┬─dhclient
                             └─2*[{NetworkManager}]
├─accounts-daemon——2*[{accounts-daemon}]
├─acpid
├─apache2——2*[apache2——26*[{apache2}]]
├─avahi-daemon——avahi-daemon
├─boltd——2*[{boltd}]
├─colord——2*[{colord}]
├─cron
├─cups-browsed——2*[{cups-browsed}]
├─cupsd——2*[dbus]
├─dbus-daemon
├─fwupd——4*[{fwupd}]
└─gdm3—┬─gdm-session-wor—┬─gdm-x-session—┬─Xorg——{Xorg}
```

# Gestionnaire des tâches Windows



The screenshot shows the Windows Task Manager window with the 'Performance' tab selected. The window title is 'Gestionnaire des tâches'. The menu bar includes 'Fichier', 'Options', and 'Affichage'. The tabs at the top are 'Processus', 'Performance', 'Historique des applications', 'Démarrage', 'Utilisateurs', 'Détails', and 'Services'. The 'Performance' tab displays a table of system resource usage.

Nom	Statut	0% Processeur	35% Mémoire	2% Disque	0% Réseau
VMware USB Arbitration Service		0%	1,1 Mo	0 Mo/s	0 Mbits/s
VMware View client USB service (32-bit) (32 bits)		0%	1,3 Mo	0 Mo/s	0 Mbits/s
VMware View Framework Node Manager (32 bits)		0%	1,4 Mo	0 Mo/s	0 Mbits/s
Windows Driver Foundation - Processeur hôte de l'infrastructure de pilotes en mode utilisateur		0%	1,1 Mo	0 Mo/s	0 Mbits/s
WMI Provider Host		0%	9,1 Mo	0 Mo/s	0 Mbits/s
<b>Processus Windows (24)</b>					
Antimalware Service Executable		0%	40,1 Mo	0 Mo/s	0 Mbits/s
Application d'ouverture de session Windows		0%	0,9 Mo	0 Mo/s	0 Mbits/s
Application de démarrage de Windows		0%	0,5 Mo	0 Mo/s	0 Mbits/s
Applications Services et Contrôleur		0%	3,5 Mo	0 Mo/s	0 Mbits/s
Gestionnaire de fenêtres du Bureau		0,1%	73,3 Mo	0 Mo/s	0 Mbits/s

At the bottom left, there is a button 'Moins de détails' with an upward arrow icon. At the bottom right, there is a button 'Fin de tâche'.

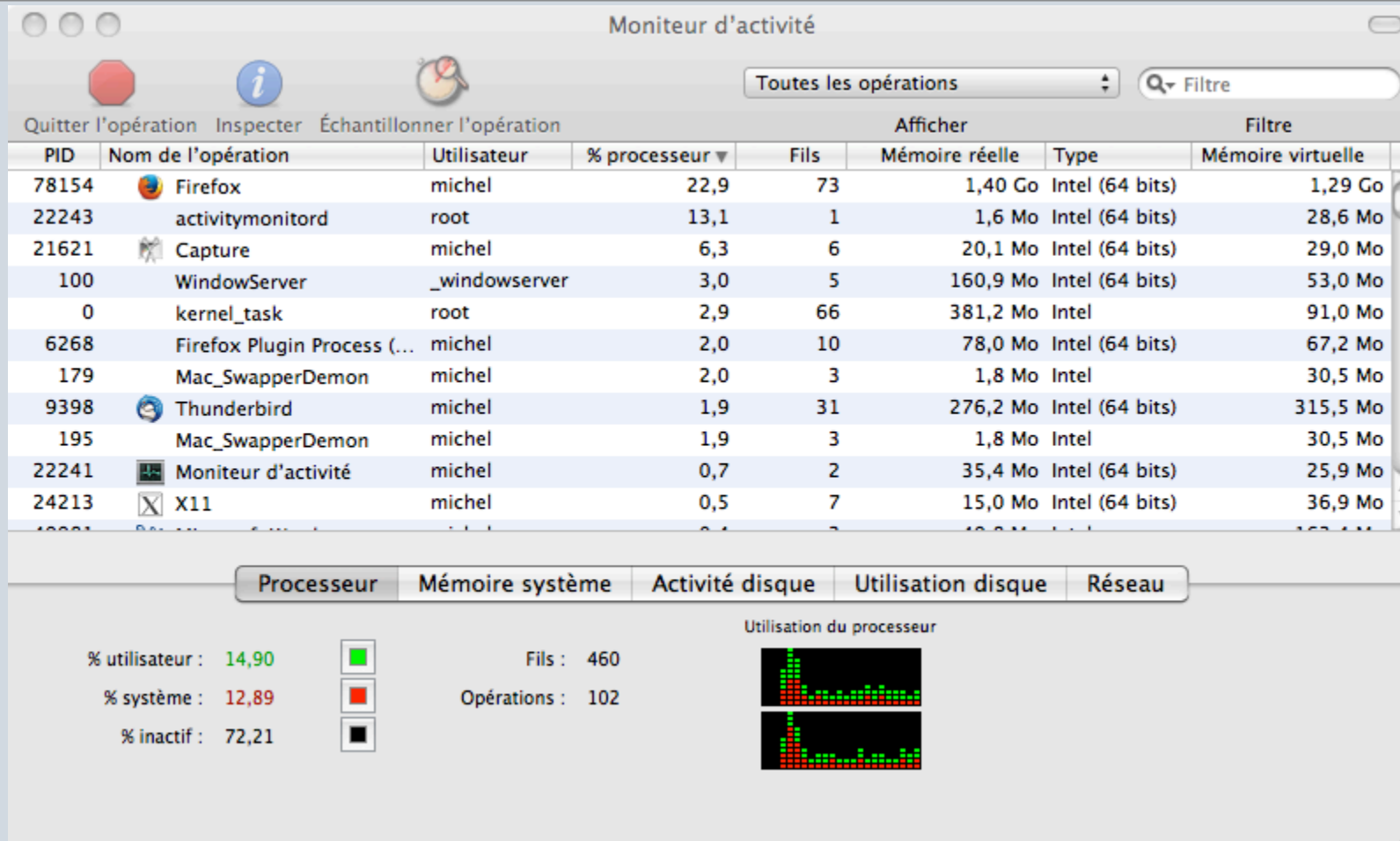
# Commande top Unix

```
lindmz.unice.fr - PuTTY
top - 22:13:41 up 46 days, 0 min, 2 users, load average: 0,00, 0,01, 0,05
Tasks: 105 total, 1 running, 104 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0,0 us, 0,0 sy, 0,0 ni,100,0 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
KiB Mem: 1027024 total, 966560 used, 60464 free, 73184 buffers
KiB Swap: 1951740 total, 7100 used, 1944640 free, 655636 cached

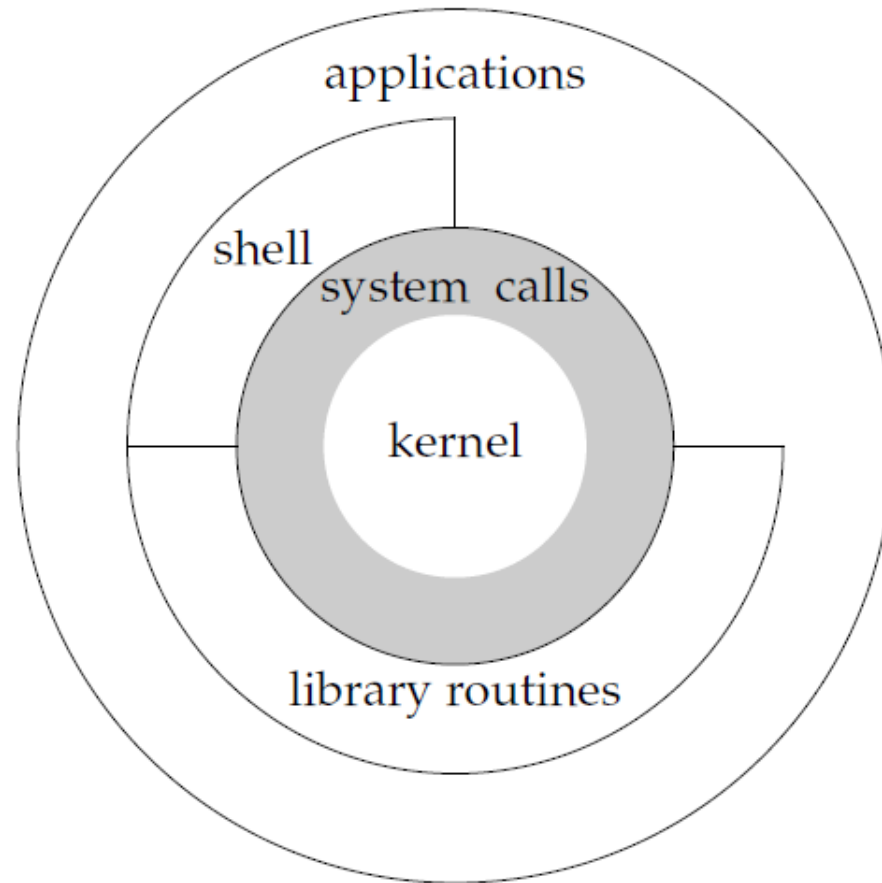
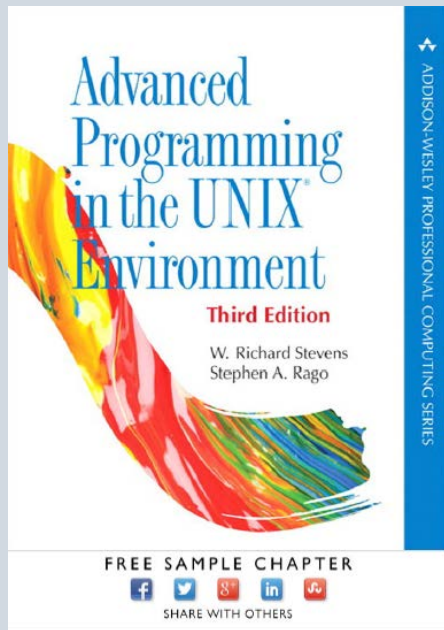

```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1	root	20	0	10648	660	628	S	0,0	0,1	0:33.34	init
2	root	20	0	0	0	0	S	0,0	0,0	0:00.06	kthreadd
3	root	20	0	0	0	0	S	0,0	0,0	0:04.42	ksoftirqd/0
5	root	20	0	0	0	0	S	0,0	0,0	0:00.00	kworker/u:0
6	root	rt	0	0	0	0	S	0,0	0,0	0:00.00	migration/0
7	root	rt	0	0	0	0	S	0,0	0,0	0:18.31	watchdog/0
8	root	0	-20	0	0	0	S	0,0	0,0	0:00.00	cpuset
9	root	0	-20	0	0	0	S	0,0	0,0	0:00.00	khelper
10	root	20	0	0	0	0	S	0,0	0,0	0:00.00	kdevtmpfs
11	root	0	-20	0	0	0	S	0,0	0,0	0:00.00	netns

# Moniteur d'Activité Mac OS X



# Architecture d'un système Unix (APUE)





# La libc

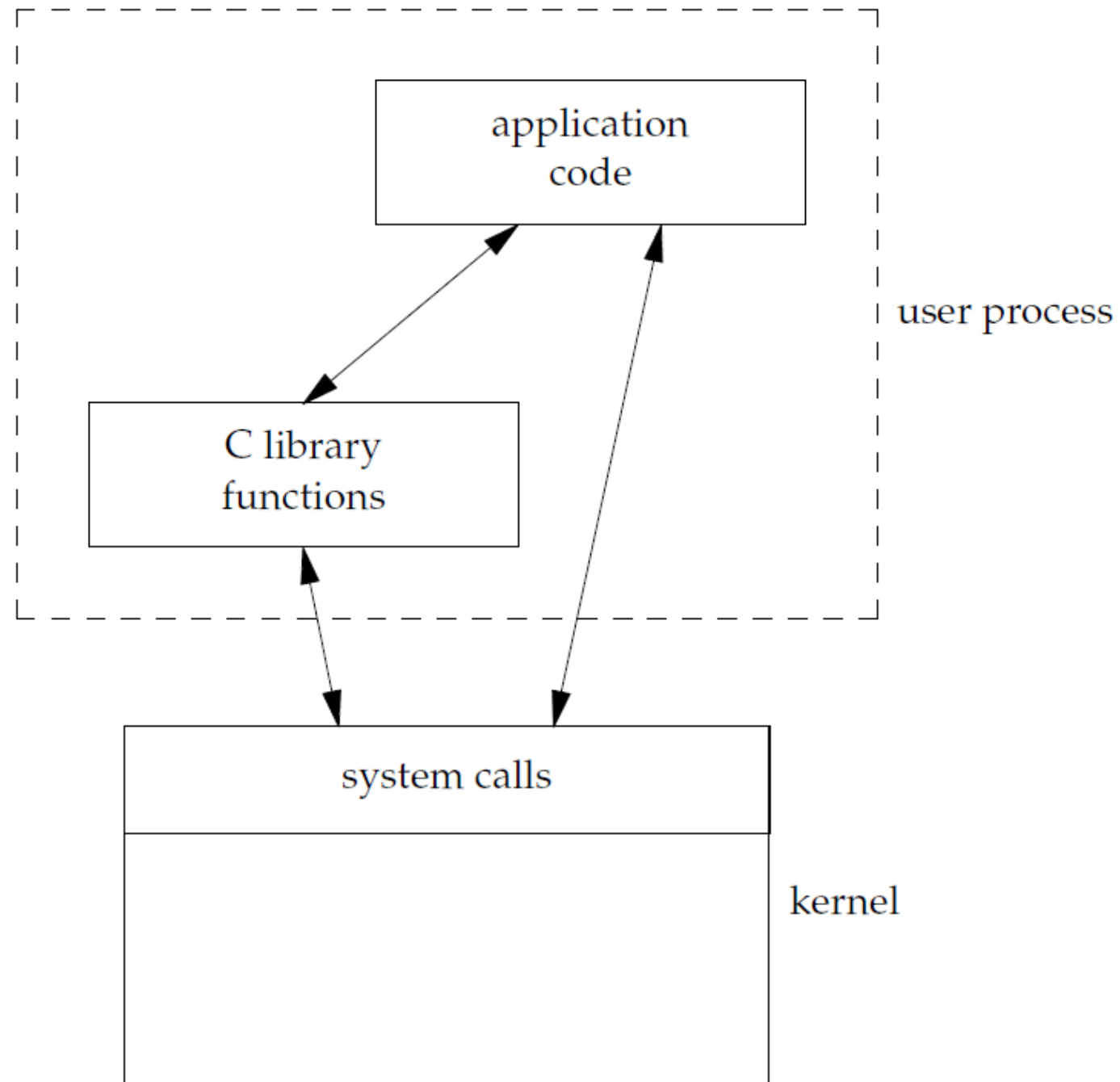
---

`/lib/x86_64-linux-gnu/libc.so.6`

`uname -a`

<http://www.gnu.org/software/libc>

- appels systèmes (man 2)
- routines C (man 3)

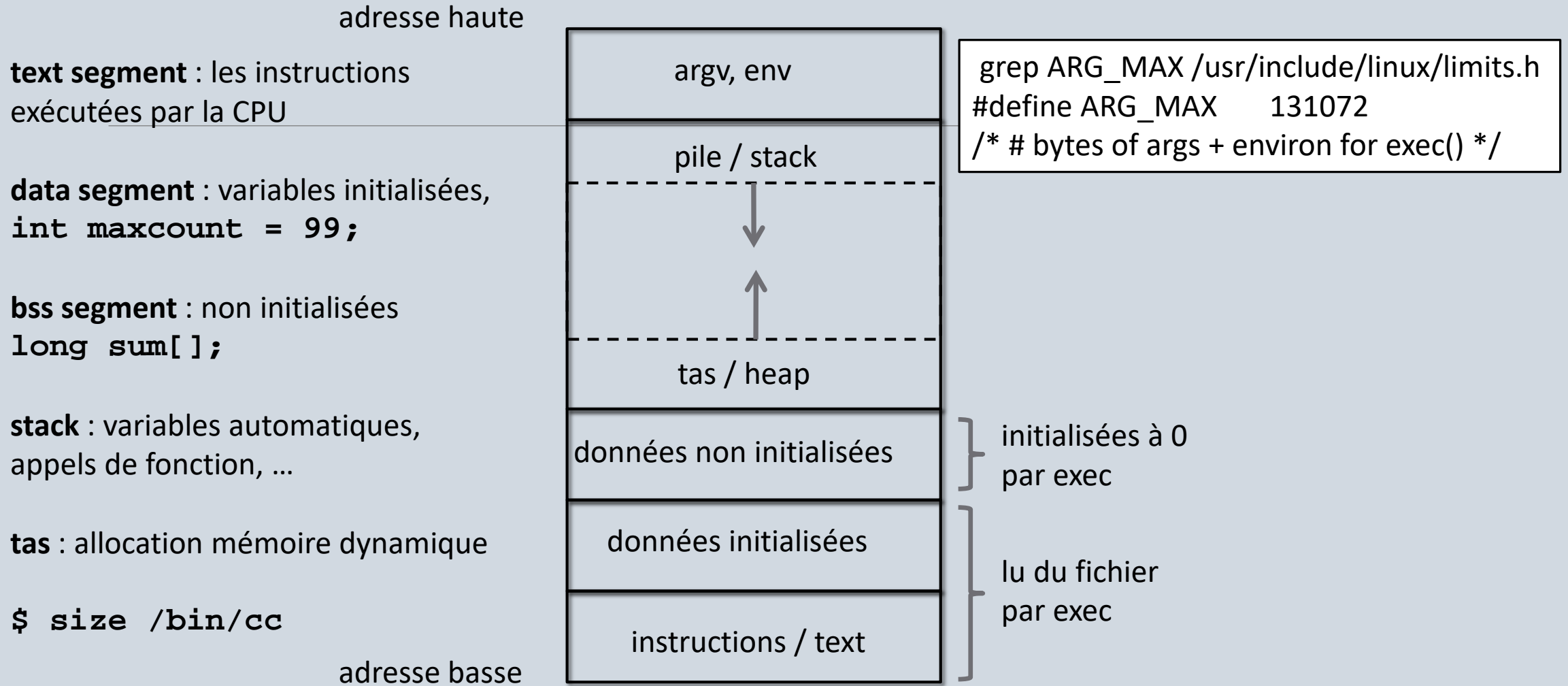


# Exécution d'un programme et appels systèmes

```
$ strace ./echoarg un deux trois
execve("./echoarg", [ "./echoarg", "un", "deux", "trois" ], [ /* 16 vars */ ]) = 0
brk(0)                                = 0x926000
access("/etc/ld.so.nohwcap", F_OK)    = -1 ENOENT (No such file or directory)
mmap(NULL, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7ffa3b91c000
...
write(1, "argument 1: un\n", 15argument 1: un
)    = 15
write(1, "argument 2: deux\n", 17argument 2: deux
)    = 17
write(1, "argument 3: trois\n", 18argument 3: trois
)    = 18
exit_group(0)                        = ?
$
```



# Organisation mémoire d'un programme C:



APUE pages 204-207

# Arguments de la ligne de commande:

```
#include <stdio.h>
#include <stdlib.h>
int main (int argc, char *argv [])
{
    for (int i = 1; i < argc; i++)
        printf ("argument %d: %s\n",
i, argv[i]);
    return EXIT_SUCCESS;
}
```

```
C:\Users\syska\Documents\M311>cl /TP echoargwin.c
Compilateur d'optimisation Microsoft (R) C/C++
version 17.00.61030 pour x86
Copyright (C) Microsoft Corporation. Tous droits
réservés.
```

```
echoargwin.c
Microsoft (R) Incremental Linker Version 11.00.61030.0
Copyright (C) Microsoft Corporation. All rights reserved.
```

```
/out:echoargwin.exe
echoargwin.obj
```

```
C:\Users\syska\Documents\M311>echoargwin.exe un
deux trois quatre etc
```

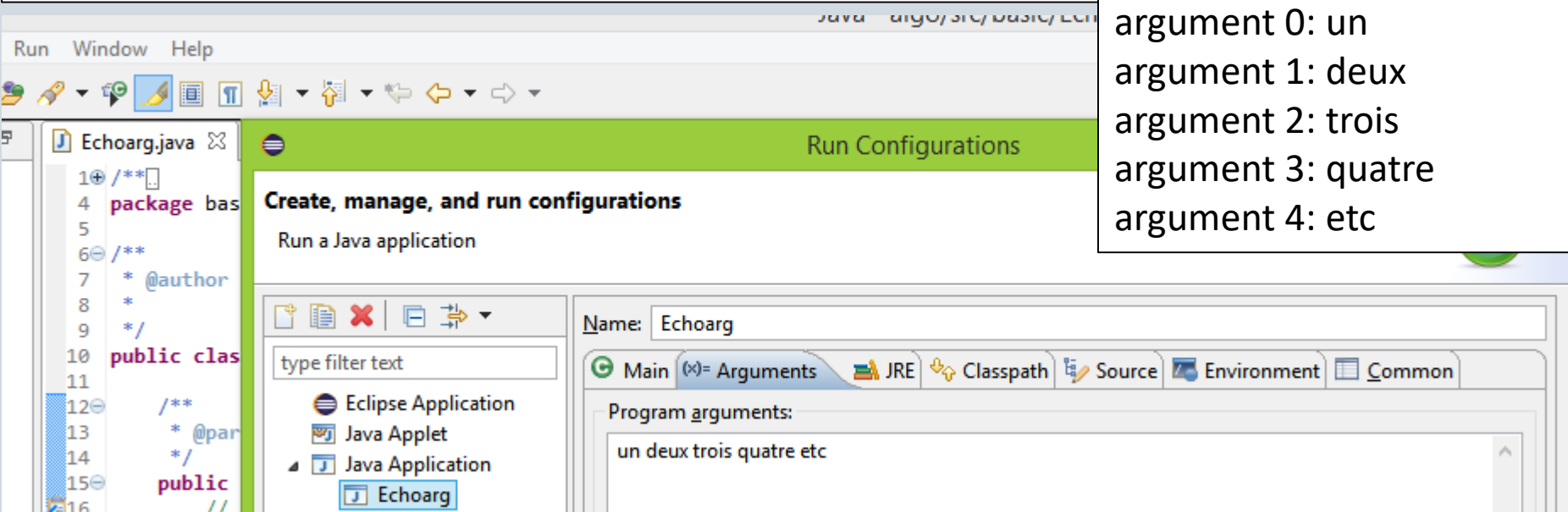
```
argument 1: un
argument 2: deux
argument 3: trois
argument 4: quatre
argument 5: etc
```

```
C:\Users\syska\Documents\M311>
```

# Arguments de la ligne de commande: Java

```
public class Echoarg {  
    public static void main(String[] args) {  
        // affiche les arguments de la ligne de commande  
        for (int i = 0; i < args.length; i++) {  
            System.out.println("argument " + i + ": " + args[i]);  
        }  
    }  
}
```

```
>"c:\Program Files\Java\jdk1.8.0_20\bin\javac" Echoarg.java  
>java Echoarg un deux trois quatre etc  
argument 0: un  
argument 1: deux  
argument 2: trois  
argument 3: quatre  
argument 4: etc
```



# Arguments de la ligne de commande: Python

```
#testArgv.py
import sys

print sys.argv

i=0
for arg in sys.argv:
    print "Argument ", i, arg
    i += 1
```

```
>c:\Python27\python.exe testArgv.py
un deux trois quatre etc
['testArgv.py', 'un', 'deux', 'trois', 'quatre', 'etc']
Argument 0 testArgv.py
Argument 1 un
Argument 2 deux
Argument 3 trois
Argument 4 quatre
Argument 5 etc
```

```
#Python 3 : list comprehension
[print("Argument", i, ": ", arg) for i, arg in enumerate(sys.argv)]
```

# Variables d'environnement

```
#include <stdio.h>
#include <stdlib.h>

int main (int argc, char *argv [])
{
    for (int i = 1; i < argc; i++)
    {
        printf ("argument %d: %s\n", i, argv[i]);
        printf ("\n%s\n\n", getenv (argv[i]));
    }
    return EXIT_SUCCESS;
}
```

```
C:\Users\syska\Documents\M311>echo argenvwin USERNAME PATH LIB
argument 1: USERNAME
```

syska

argument 2: PATH

C:\Program Files (x86)\Microsoft Visual Studio 11.0\Common7\IDE\CommonExtensions\Microsoft Performance Toolkit\;C:\Program Files (x86)\Intel\OpenCL SDK\3.0\bin\x86;C:\Program Files (x86)\Intel\OpenCL SDK\3.0\bin\x86\_64

argument 3: LIB

C:\Program Files (x86)\Microsoft Visual Studio 11.0\VC\LIB;C:\Program Files (x86)\Microsoft

C:\Users\syska\Documents\M311&gt;

# Tout l'environnement (Windows)

```
/* d'après http://tenouk.com/cpluspluscodesnippet/listsystemenvvariables.html */  
#include <stdio.h>  
#include <stdlib.h>  
#include <windows.h>  
  
int main (int argc, char *argv []){  
    LPTSTR lpszVariable;  
    LPVOID lpvEnv = GetEnvironmentStrings();  
    for (lpszVariable = (LPTSTR) lpvEnv; *lpszVariable; lpszVariable++){  
        while (*lpszVariable)  
            putchar(*lpszVariable++);  
        putchar('\n');  
    }  
    char buffer[65535];  
    GetEnvironmentVariableA( "PATH", buffer, 65535 );  
    printf ("\n<<%s>>\n\n", buffer);  
    return EXIT_SUCCESS;  
}
```

Équivalent aux commandes  
SET  
et  
ECHO %PATH%  
dans un CMD.EXE



# Tout l'environnement (Unix)

```
/* APUE Stevens */
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]) {
    int i;
    char **ptr;
    extern char **environ; /* this is a Unixism; according to Posix, it should be declared in <unistd.h>.
        See http://www.unix.org/single\_unix\_specification, See also getenv() */
    // https://stackoverflow.com/questions/1433204/how-do-i-use-extern-to-share-variables-between-source-files
    for (i = 0; i < argc; i++) /* echo all command-line args */
        printf("argv[%d]: %s\n", i, argv[i]);

    for (ptr = environ; *ptr != 0; ptr++) /* and all env strings */
        printf("%s\n", *ptr);

    exit(EXIT_SUCCESS);
}
```

# Tout l'environnement (Unix)

---

```
// On peut obtenir les valeurs des variables d'environnement dans un
// tableau envp similaire à argv
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[], char *envp[]){
    int i = 0;
    for (i = 0; envp[i] != NULL; i++)
        fprintf(stdout, "%d : %s\n", i, envp[i]);
    return EXIT_SUCCESS;
}
```



# Compilation d'un programme C / Linux

---

compilateur GNU GCC:

```
gcc echoarg.c -o echoarg
```

```
./echoarg un deux trois quatre etc
```

Par défaut (si on ne spécifie pas -o executable) le résultat de la compilation est le fichier a.out

# Compilation et Makefile

Introduction à la commande make:

- un fichier Makefile contient des règles
- la commande make exécute les règles contenues dans Makefile
- Exemples
  - make
  - make all
  - make clean

Permet de gérer les dépendances dans la construction d'un projet

Ne recompile que les fichiers sources utiles

```
CFLAGS = -Wall -g -std=gnu99
```

```
EXECUTABLES = echoall \  
              echoarg \  
              getenv
```

```
all : ${EXECUTABLES}
```

```
clean :
```

```
@rm -f core *~ *.o *.out  
@rm -f ${EXECUTABLES}
```



# Compilation et Makefile

---

<http://gl.developpez.com/tutoriel/outil/makefile/>

<http://www.gnu.org/software/make/>

Voir aussi

<http://ant.apache.org/>

<http://maven.apache.org/>

<http://www.cmake.org/>