Unix et Programmation Shell

Philippe Langevin

département d'informatique
UFR sciences et technique
université du sud Toulon Var

Automne 2013

document

brouillon en révision

- site du cours :
  http://langevin.univ-tln.fr/cours/UPS/upsh.html
- localisation du fichier :
  http://langevin.univ-tln.fr/cours/UPS/doc/intro.pdf

dernières modifications

```
intro.tex 2013-06-23 10:07:23.697660496 +0200
macros.tex 2013-06-22 23:42:16.868263946 +0200
prologue.tex 2013-06-22 22:10:45.967471216 +0200
upsh.tex 2013-06-22 11:24:27.941744751 +0200
tools.tex 2013-06-22 11:24:27.940741957 +0200
term.tex 2013-06-22 11:24:27.934741399 +0200
syntaxe.tex 2013-06-22 11:24:27.931741678 +0200
proc.tex 2013-06-22 11:24:27.927741678 +0200
piped.tex 2013-06-22 11:24:27.922741399 +0200
perm.tex 2013-06-22 11:24:27.921741678 +0200
part.tex 2013-06-22 11:24:27.918742237 +0200
man.tex 2013-06-22 11:24:27.911741957 +0200
langage.tex 2013-06-23 10:04:58.364659987 +0200
file.tex 2013-06-22 11:24:27.896741119 +0200
dup.tex 2013-06-22 11:24:27.885741678 +0200
fic.tex 2013-06-22 11:24:27.885741678 +0200
bash.tex 2013-06-22 11:24:27.881741957 +0200
```

1 - shell unix

- origine
- unices
- GNU/linux
- distribution
- shell unix
- GUI vs CLI
- exemple

naissance de unix

Deux pionniers de la compagnie BELL LABS sont à l'origine du système unix :

naissance de unix

Deux pionniers de la compagnie BELL LABS sont à l'origine du système unix :

1969 Ken Thompson crée le système UNICS

naissance de unix

Deux pionniers de la compagnie BELL LABS sont à l'origine du système unix :

1969 Ken Thompson crée le système UNICS

1971 Dennis Ritchie crée Le langage C utile pour le développement d'unix.

naissance de unix

Deux pionniers de la compagnie BELL LABS sont à l'origine du système unix :

1969 Ken Thompson crée le système UNICS

1971 Dennis Ritchie crée Le langage C utile pour le développement d'unix.

[PSLC] : Le langage C est un bon moyen de comprendre les mécanismes mis en oeuvre au coeur du noyau, c.f. Programmation Système en Langage C sous Linux, par C. Blaess.

naissance de unix

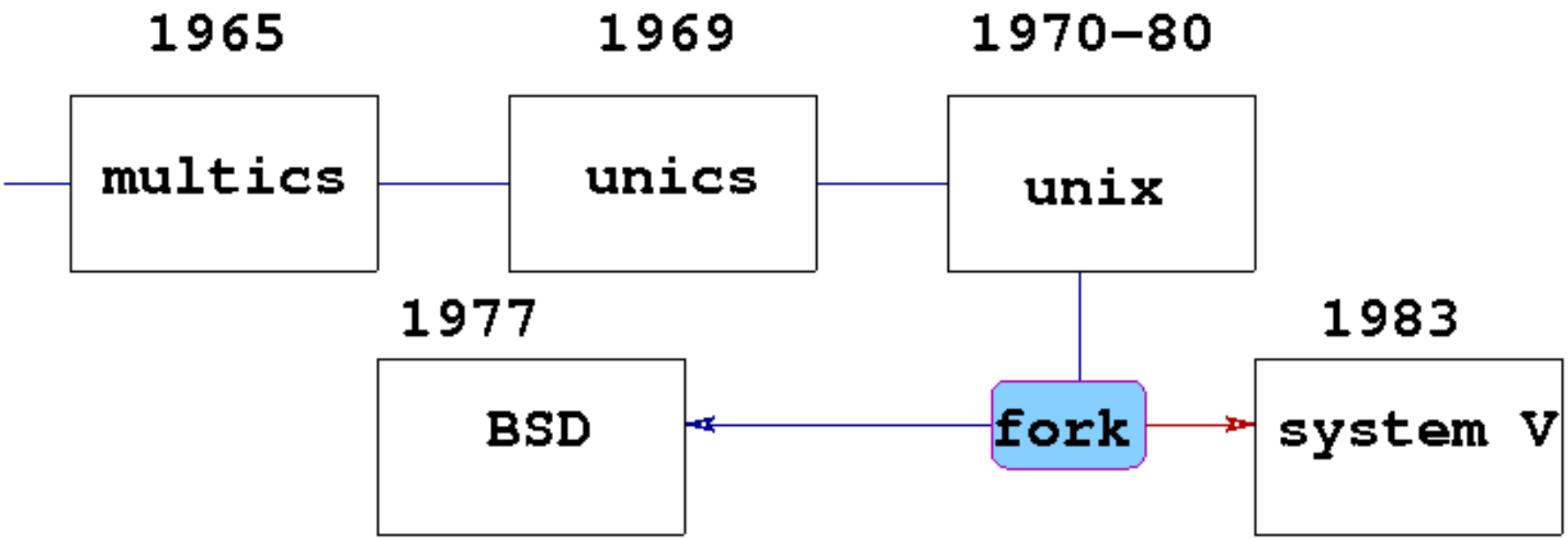
Deux pionniers de la compagnie BELL LABS sont à l'origine du système unix :

1969 Ken Thompson crée le système UNICS

1971 Dennis Ritchie crée Le langage C utile pour le développement d'unix.

[PSLC] : Le langage C est un bon moyen de comprendre les mécanismes mis en oeuvre au coeur du noyau, c.f. Programmation Système en Langage C sous Linux, par C. Blaess.
[EPI] : Le système unix et l'informatique en tant que discipline ont le même age !

chronologie



freeBSD	solaris	HP-UX
openBSD	linux	AIX
netBSD	GNU/linux	XENIX
mac OS X	android	ULTRIX

unix populaires

unix a donné naissance à une famille de systèmes, les *unices* dont les plus populaires sont :

1983	System V	Bell labs, AT&T.
1977	BSD	Berkeley Software Distribution
1990	GNU/Linux	Logiciel Libre
1999	OS X	next, apple.
2003	android	androïde, google.

L'ensemble des industriels acteurs du développemement du système unix sont regoupés dans l'opengroup propriétaire de la marque unix dont le "Single UNIX Specification" certifie les systèmes unix.

influences

Trois groupes influent sur la normalisation des systèmes `unix` :

- POSIX : Portable Operating System Interface (IEEE).
- BSD
- GNU : Gnu is Not Unix, logiciel libre.

Je vous recommande la description du projet GNU par R. Stallman, et la lecture de la cathédrale et le bazar par E. Raymond.

dialecte

```
PS(1)    Linux User's Manual
NAME
   ps - report a snapshot of the current processes.

DESCRIPTION
   ps displays information about a selection of the
active processes. If you want a repetitive update of
the selection and the displayed information, use top.

This version of ps accepts several kinds of options:
 1 UNIX options,   must be preceded by a dash.
 2 BSD options,    must not be used with a dash.
 3 GNU long options, preceded by two dashes.
Options of different types may be freely mixed,
                          but conflicts can appear.
```

GNU/linux

Dans les salles de travaux-pratiques, vous utiliserez un système d'exploitation GNU/linux, fusion de deux composantes du logiciel libre :

- noyau linux (Linus Torvalds, 1991),
- utilitaires GNU (Richard Stallman, 1983).

Plus précisément, une distribution ubuntu, basée sur debian. Il s'agit d'un environnement de travail unix de qualité directement issu du logiciel libre.

- hurd : le noyau GNU n'est pas encore opérationnel.

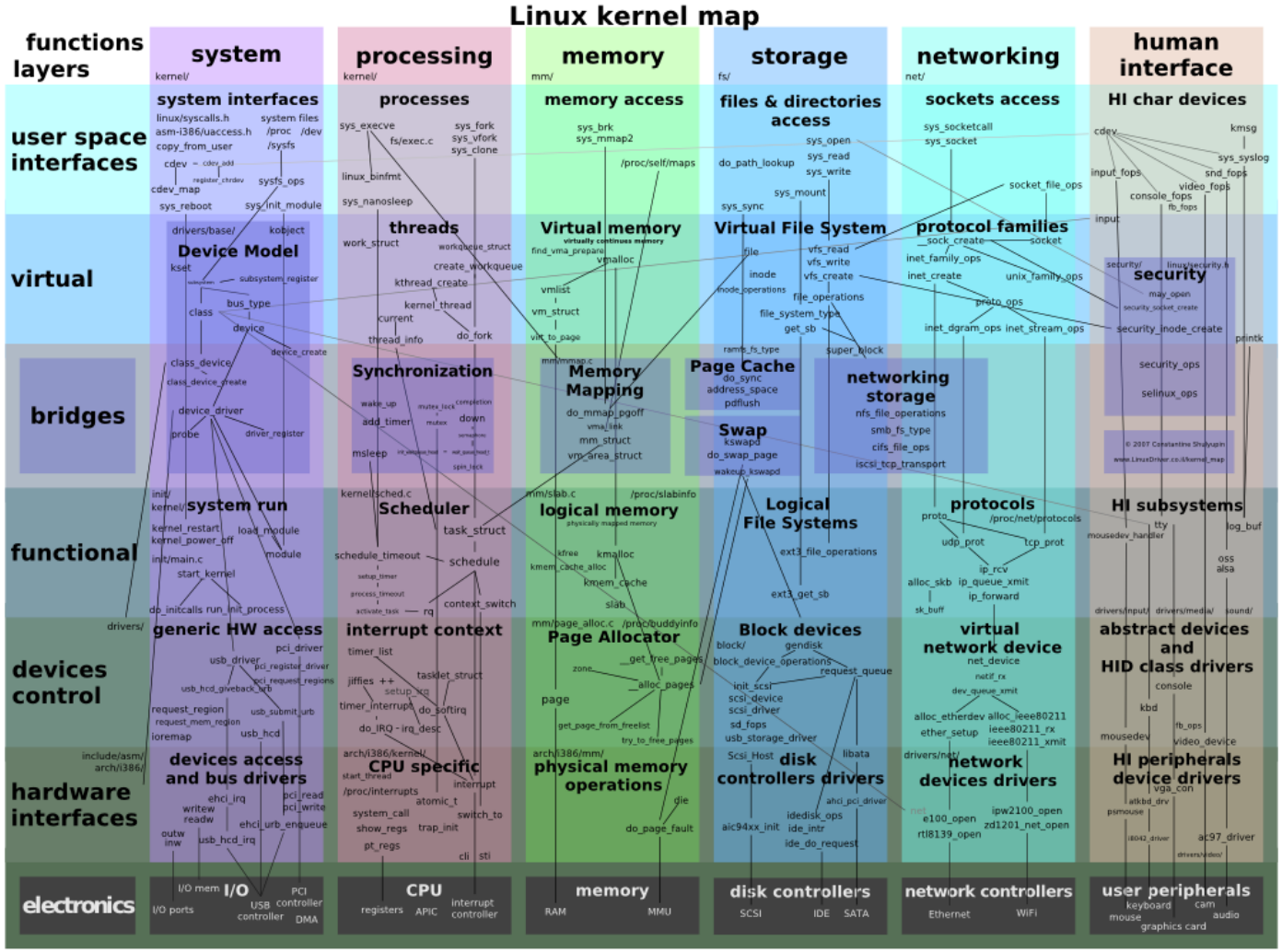
paquets gnu (2012)

a2ps acct acm adns aetherspace alive anubis archimedes aris aspell auctex autoconf autoconf-archive autogen automake avl ballandpaddle barcode bash bayonne bazaar bc bfd binutils bison bool bpel2owfn c-graph ccaudio ccide ccrtp ccscript cflow cgicc chess cim classpath classpathx clisp cobol combine commoncpp complexity config coreutils cpio cppi cssc dap dc ddd ddrescue dejagnu denemo dia dico diction diffutils dionysus dismal djgpp dmd dominion dotgnu dotgnu-forum dotgnu-pnet dr-geo ed edma electric emacs emacs-muse emms enscript eprints epsilon fdisk ferret findutils fontutils freedink freefont freeipmi freetalk fribidi gama garpd gawk gcal gcc gcide gcl gcompris gdb gdbm gengen gengetopt gettext gforth ggradebook ghostscript gift gimp gleem glib global glpk glue gmediaserver gmorph gmp gnash gnat gnats gnatsweb gnome gnowsys gnu-arch gnu-c-manual gnu-crypto

gnuae gnubatch gnubg gnubiff gnubik gnucap gnucash gnucomm gnue gnufm gnugo gnuit gnujdoc gnujump gnukart gnulib gnumach gnumed gnumeric gnump3d gnun gnunet gnupg gnupod gnuprologjava gnuradio gnurobots gnuschool gnushogi gnuskies gnusound gnuspeech gnuspool gnustandards gnustep gnutls gnutrition gnuzilla goptical gorm gpaint gperf gprolog grabcomics greg grep gretl groff grub gsasl gsegrafix gsl gsrc gss gtick gtk+ gtypist guile guile-dbi guile-gnome guile-gtk guile-ncurses guile-rpc gurgle gv gvpe gxmessage gzip halifax health hello help2man hp2xx httptunnel hurd hyperbole icecat idutils ignuit indent inetutils intlfonts jacal java-getopt jdresolve jel jwhois kawa kopi leg less libc libcdio libextractor libffcall libgcrypt libiconv libidn libmatheval libmicrohttpd libredwg librejs libsigsegv libtasn1 libtool libunistring libxmi lightning lilypond linux-libre liquidwar6 lispintro lrzsz lsh m4 macchanger mailman mailutils

make marst maverik mc mcron mcsim mdk mediagoblin melting metaexchange metahtml mifluz mig miscfiles mit-scheme moe motti mpc mpfr mtools myserver nana nano ncurses nettle network ocrad octave oleo orgadoc osip packaging panorama paperclips parallel parted pascal patch paxutils pcb pdf pem pexec pgccfd phantom-home phpgroupware pies pipo plotutils polyxmass powerguru proxyknife pspp psychosynth pth pythonwebkit qexo quickthreads radius rcs readline recutils reftex rottlog rpge rush sather scm screen sed serveez sharutils shishi shmm shtool sipwitch slib smalltalk smarteiffel snakecharmer social solfege sourceinstall spacechart speex spell sqltutor src-highlite stalkerfs stow stump superopt swbis sysutils talkfilters tar termcap termutils teseq teximpatient texinfo texmacs thales time tramp trans-coord trueprint units unrtf userv uucp vc-changelog vc-dwim vcdimager vera vmgen vmslib w3 wb wdiff websocket4j webstump wget which womb xaos xboard xhippo xlogmaster xmlat xnee xorriso zile

noyau linux


Linux kernel map
functions layers
system
processing
memory
storage
networking
human interface
user space interfaces
virtual
bridges
functional
devices control
hardware interfaces
electronics
system interfaces
processes
memory access
files & directories access
sockets access
HI char devices
threads
Virtual memory
Virtual File System
protocol families
Device Model
security
Synchronization
Memory Mapping
Page Cache
networking storage
Swap
system run
Scheduler
logical memory
Logical File Systems
protocols
HI subsystems
generic HW access
interrupt context
Page Allocator
Block devices
virtual network device
abstract devices and HID class drivers
devices access and bus drivers
CPU specific
physical memory operations
disk controllers drivers
network devices drivers
HI peripherals device drivers
I/O
CPU
memory
disk controllers
network controllers
user peripherals

distribution

distribution

Le système GNU/linux est mis en forme au sein de plusieurs distributions qui intègrent le noyau et les utilitaires avec

- une politique de distribution,
- un système de maintenance,
- une communauté

qui les caractérisent.

```
⇝ uname -mors
Linux 2.6.34.9-69.fc13.i686 i686 GNU/Linux
⇝ ssh pl@192.168.0.150 uname -mors
Linux 3.8.13-100.fc17.i686.PAE i686 GNU/Linux
⇝  ssh pl@ouaib.univ-tln.fr uname -mors
Linux 2.6.18-53.1.4.el5 i686 GNU/Linux
⇝ ssh pl@imath01
Linux 2.6.18-348.4.1.el5.centos.plus x86_64 GNU/Linux
```

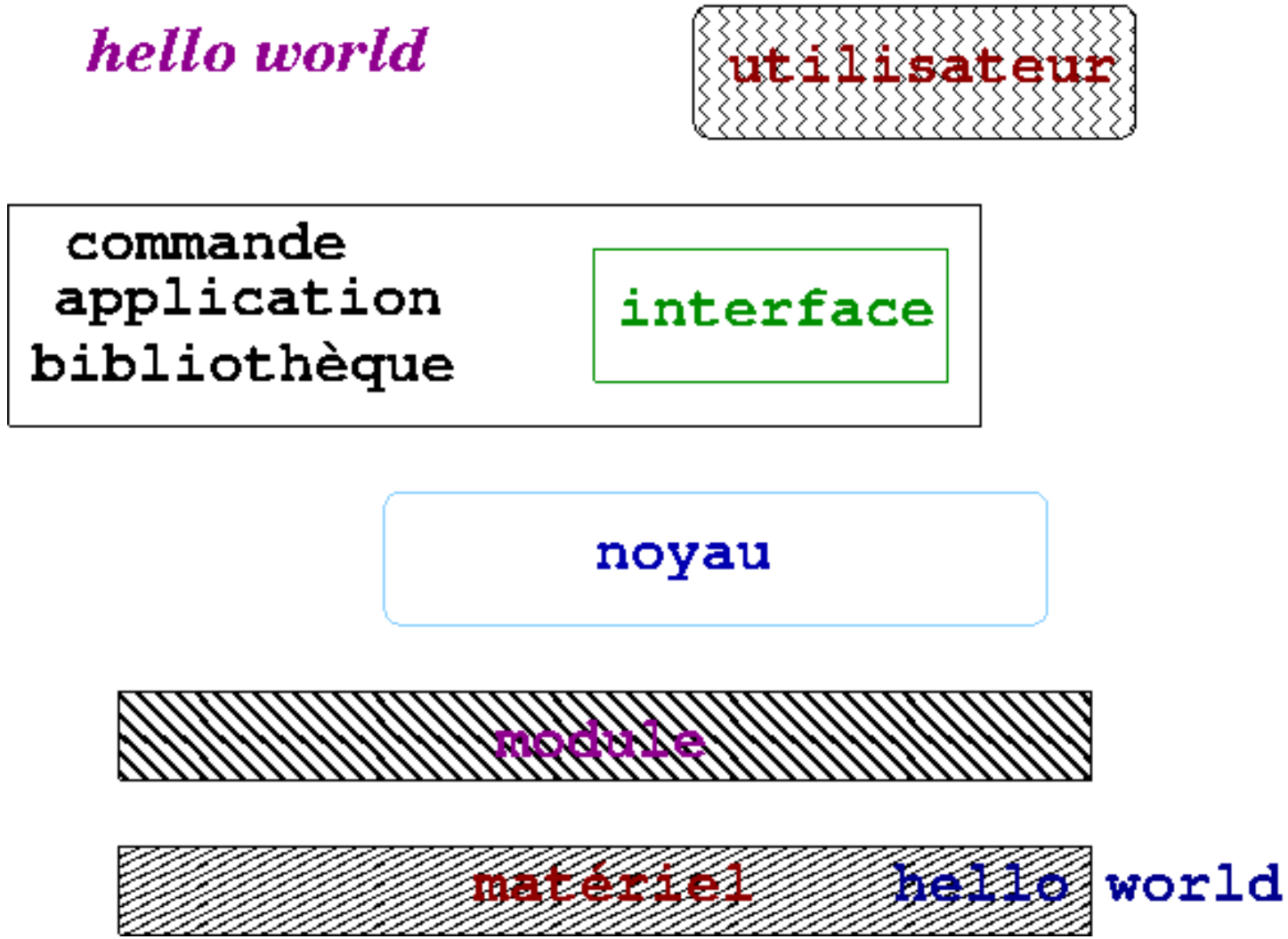
shell ?

La terminologie de la communauté `unix` comprend quelques bizarreries, sigles et acronymes plus ou moins célèbres :

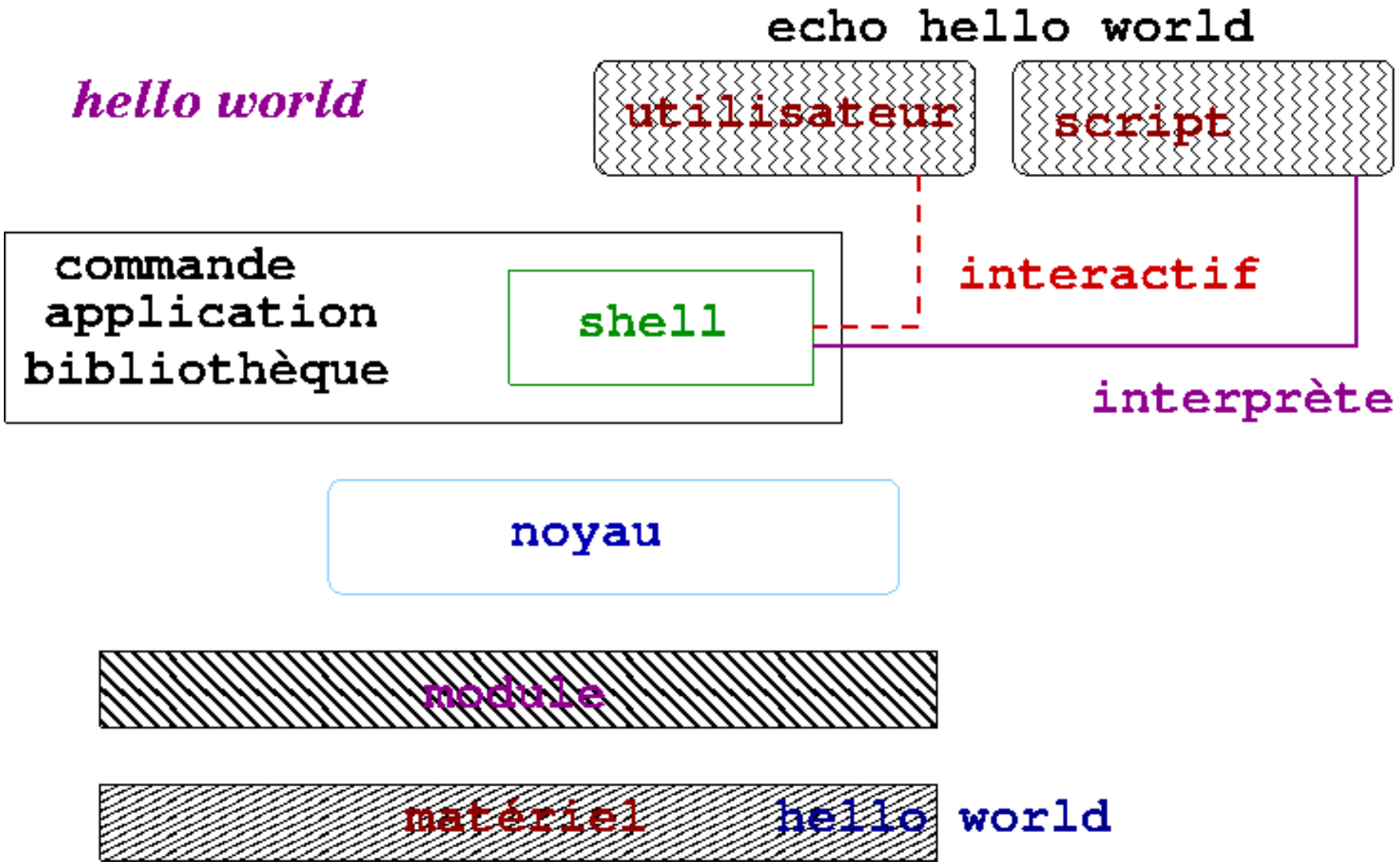
- Portable Operating System Interface X
- foo, bar ? ?
- biff
- shell ? bash ! shabang ♯ !

[RFC-3092] International Engineering Task Force.
[FAQ] : `unix` Frequently Asked Questions.

système d'exploitation



système d'exploitation



Thompson shell

La notion de `shell` apparaît dans le système MULTICS, il s'agit d'une application à l'interface entre le noyau et les utilitaires. Le shell développé au début des années 70 par Thompson est un interprète de commandes qui introduit la syntaxe des deux opérateurs fondamentaux des systèmes `unix`, la *redirection* :

$$\texttt{commande} > \textit{destination} < \textit{source}$$

et le *pipeline* de Douglas McIlroy

$$\texttt{commande} \mid \texttt{commande}$$

[man 1 sh]

trace

```
ltrace -f -o this.$$  echo hello world
cut -c1-50  <this.$$  >hello-lib.out
strace -f -o this.$$ echo hello world
cut -c1-50  <this.$$  >hello-sys.out
```

echo hello world

```
3338 __libc_start_main(0x8048d50, 3, 0xbfb26c44, 0
3338 getenv("POSIXLY_CORRECT")                      =
3338 strrchr("echo", '/')                           =
3338 setlocale(LC_ALL, "")                          =
3338 bindtextdomain("coreutils", "/usr/share/local
3338 textdomain("coreutils")                        =
3338 __cxa_atexit(0x80498a0, 0, 0, 0x804eff4)       =
3338 fputs_unlocked(0xbfb274a8, 0x415c3a00, 0x7b70
3338 fputs_unlocked(0xbfb274ae, 0x415c3a00, 0x7b70
3338 exit(0 <unfinished ...>
3338 __fpending(0x415c3a00, 0x415c2ff4, 0xb7709000
3338 fileno(0x415c3a00)                             =
3338 __freading(0x415c3a00, 0x84f3848, 10000, 0x41
3338 __freading(0x415c3a00, 0xe70bf85, 0x51c5beaa,
3338 fflush(0x415c3a00)                             =
3338 fclose(0x415c3a00)                             =
```

echo hello world

```
3344  execve("/bin/echo", ["echo", "hello", "world
3344  brk(0)                                    = 0x979a00
3344  mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_
3344  access("/etc/ld.so.preload", R_OK) = -1 ENOE
3344  open("/etc/ld.so.cache", O_RDONLY|O_CLOEXEC)
3344  fstat64(3, {st_mode=S_IFREG|0644, st_size=79
3344  mmap2(NULL, 79326, PROT_READ, MAP_PRIVATE, 3
3344  close(3)                                  = 0
3344  open("/lib/libc.so.6", O_RDONLY|O_CLOEXEC) =
3344  read(3, "\177ELF\1\1\1\3\0\0\0\0\0\0\0\0\3\0
3344  fstat64(3, {st_mode=S_IFREG|0755, st_size=20
3344  mmap2(0x41415000, 1776316, PROT_READ|PROT_EX
3344  mprotect(0x415c0000, 4096, PROT_NONE) = 0
3344  mmap2(0x415c1000, 12288, PROT_READ|PROT_WRIT
3344  mmap2(0x415c4000, 10940, PROT_READ|PROT_WRIT
3344  close(3)                                  = 0
```

Bourne [Again] shell

Le Bourne shell est introduit dans le système UNIX V7. Il a été développé par Stephen Bourne avec l'objectif de créer un outil de scriptage des commandes pour faciliter l'administration du système. `sh` est devenu populaire pour sa facilité d'emploi et sa rapidité, standard des systèmes UNIX, c'est toujours le shell par défaut du compte `root`, de certaines commandes `make`.

Bourne [Again] shell

Le Bourne shell est introduit dans le système UNIX V7. Il a été développé par Stephen Bourne avec l'objectif de créer un outil de scriptage des commandes pour faciliter l'administration du système. `sh` est devenu populaire pour sa facilité d'emploi et sa rapidité, standard des systèmes UNIX, c'est toujours le shell par défaut du compte `root`, de certaines commandes `make`.

Le shell de base `sh`

Les `alias`, l'historique des commandes et le contrôle des processus ne sont pas gérés par `sh`.

Le Bourne Again shell est un projet GNU `bash` démarré en 1980 par Brian Fox, actuellement maintenu par Chet Ramey.

shell par défaut

```
⇝ /bin/sh
sh-4.1$ exit
exit
⇝ which sh
/bin/sh
⇝ ls -l /bin/sh
lrwxrwxrwx. 1 root root 4 9 aout 08:47 /bin/sh -> bash
⇝ echo $SHELL
/bin/bash
⇝ echo -e 'all:\n\techo $(SHELL)' > \
make: entrant dans le repertoire /tmp
echo /bin/sh
/bin/sh
make: quittant le repertoire /tmp
```

shell populaires

Les shells font légion.

shell	shabang
`sh`	6 490 000
`bash`	5 380 000
`ash`	345 000
`ksh`	323 000
`csh`	254 000
`tcsh`	116 000
`zsh`	86 000

TAB.: popularité de quelques shells

Ces slides concernent mon shell préféré `bash`...

Aprés le lambis bien entendu !

[comparatif des shells]

Interfaces

L'utilisateur d'un ordinateur interagit avec le système d'exploitation au moyen d'une interface graphique ou bien d'une interface texte.

- Graphic User Interface
- Command Line Interface

La notion de `shell` est étendue à l'ensemble des interfaces y compris graphique :`gnome`, `kde`, `lxde`, `xfce`, `awesome` ...

- L'évolution des GUIs fait souvent débat !
- Le monde des CLIs parait plus zen...

suggestion : essayez les interfaces !

CLI vs GUI

- Le mode graphique se veut intuitif et facile d'emploi.
- Le mode textuel est puissant mais peu intuitif.

Il n'y a pas lieu d'opposer les deux modes qui sont parfaitement complémentaires, les tâches sur les fichiers textes sont plus faciles à réaliser par la console.
En général, lors d'une session graphique, les utilisateurs ouvrent des pseudos terminaux pour effectuer certaines tâches avec l'interprète de commande.

C-production

Evaluer une C-production

combien de lignes de codes C dans le répertoire `./CC` ?

C-production

Evaluer une C-production

combien de lignes de codes C dans le répertoire `./CC` ?

Pas de solution graphique !

Difficile d'interpréter cette question par des mouvements de souris...

C-production

Evaluer une C-production

combien de lignes de codes C dans le répertoire `./CC` ?

Pas de solution graphique !

Difficile d'interpréter cette question par des mouvements de souris...

Plusieurs solutions textuelles

Au fil des années, les applications `unix` ont été développées, améliorées pour répondre efficacement ce type de questions.

Une ligne de commandes

Par exemple :

```
$ find ~/CC -name "*.c" -exec wc -l {} \;
| awk 'BEGIN {s=0} {s=s+$0} END {print s}'

400942
```

Une ligne de commandes

Par exemple :

```
$ find ~/CC -name "*.c" -exec wc -l {} \;
| awk 'BEGIN {s=0} {s=s+$0} END {print s}'

400942
```

Une solution facile à comprendre plus difficile à reproduire sans connaître les usages des commandes et arguments passés par la ligne de commande.

- `find` : outil pour la recherche de fichiers.
- `wc` : compter les lignes, mots, octets.
- `awk` : commande d'Alfred Aho, Peter Weinberger et Brian Kernighan pour filtrer les lignes d'un fichier texte.

script

Le rôle d'un informaticien est de réaliser des travaux avec une machine, le plus souvent, dans un temps limité :-).
Il convient déviter des pertes de temps qui ont pour origine

- erreurs
- lacunes

Un *script* est un fichier de commandes pour le shell. La première ligne du script peut préciser le shell d'exécution :

```
#!/bin/myshell
...
```

c'est le fameux `shabang`.

exemple

```
#!/bin/bash
sum=0
while read num rem
  do
  let sum+=$num
  done < find $1 -name "*.$2" -exec wc -l {} \;
echo $sum
```

Aprés avoir sauver ces lignes dans un fichier `count.sh` :

```
⇝ chmod u+x count.sh
```

pour le rendre exécutable, on peut alors lancer

```
⇝ ./count.sh '~/CC' '*.c'
```

commande complexe

Avec de la pratique, il est possible de d'exécuter une tache complexe directement en ligne de commande :

```
↝ find ~ -name "*.txt" -exec wc -l {} \;
  | ( sum=0; while read num rem ;
  do let sum+=$num;  done; echo $sum) > sum.txt
```

Il s'agit d'une *commande composée* s'articulant sur des *commandes simples* incluant un *pipeline* et une *redirection*.