Soutien informatique : Unix

Sergey Kirgizov

Unix est partout !

et probablement dans vos poches.

- Installer
  - termux sur votre téléphone Android
  - OpenTerm ou LibTerm sur iPhone
- Executurer `uname -a`
- `uname` c'est diminutif pour unix name.

Unix est partout !

sur vos portables, mais aussi sur les superordinateurs les plus puissants au monde

Top 10 positions of the 55th TOP500 in June 2020[24]

Rank	Rmax Rpeak (PFLOPS)	Name	Model	Processor	Interconnect	Vendor	Site country, year	Operating system
1 ▲	415.530 513.855	Fugaku	Supercomputer Fugaku	A64FX	Tofu interconnect D	Fujitsu	RIKEN Center for Computational Science Japan, 2020	Linux (RHEL)
2 ▼	148.600 200.795	Summit	IBM Power System AC922	POWER9, Tesla V100	InfiniBand EDR	IBM	Oak Ridge National Laboratory United States, 2018	Linux (RHEL)
3 ▼	94.640 125.712	Sierra	IBM Power System S922LC	POWER9, Tesla V100	InfiniBand EDR	IBM	Lawrence Livermore National Laboratory United States, 2018	Linux (RHEL)
4 ▼	93.015 125.436	Sunway TaihuLight	Sunway MPP	SW26010	Sunway[25]	NRCPC	National Supercomputing Center in Wuxi China, 2016[25]	Linux (Raise)
5 ▼	61.445 100.679	Tianhe-2A	TH-IVB-FEP	Xeon E5–2692 v2, Matrix-2000[26]	TH Express-2	NUDT	National Supercomputing Center in Guangzhou China, 2013	Linux (Kylin)
6 ▲	35.450 51.721	HPC5	Dell	Xeon Gold 6252, Tesla V100	Mellanox HDR Infiniband	Dell EMC	Eni Italy, 2020	Linux (CentOS)
7 ▲	27.580 34.569	Selene	Nvidia	Epyc 7742, Ampere A100	Mellanox HDR Infiniband	Nvidia	Nvidia United States, 2020	Linux (Ubuntu)
8 ▼	23.516 38.746	Frontera	Dell C6420	Xeon Platinum 8280 (subsystems with e.g. POWER9 CPUs and Nvidia GPUs were added after official benchmarking[10])	InfiniBand HDR	Dell EMC	Texas Advanced Computing Center United States, 2019	Linux (CentOS)
9 ▲	21.640 29.354	Marconi-100	IBM Power System AC922	POWER9, Volta V100	Dual-rail Mellanox EDR Infiniband	IBM	CINECA Italy, 2020	Linux (RHEL)
10 ▼	21.230 27.154	Piz Daint	Cray XC50	Xeon E5-2690 v3, Tesla P100	Aries	Cray	Swiss National Supercomputing Centre Switzerland, 2016	Linux (CLE)

https://en.wikipedia.org/wiki/TOP500

Objectif d'apprentissage

Vous apprendrez comment utiliser les systèmes d'exploitation type Unix el le langage C.

Plan de cours

CM 1 — Unix, écosystème du shell

CM 2 — Langage C, structure, compilation et execution d'un programme

CM 3 — Langage C, tableaux, arithmétique des pointeurs, notions d'allocation de mémoire

5 TD

7 TP — avec Duncan Luguern, un projet à faire à la fin.

En TP vous allez créer vos propres logicielles d'Unix

Notes

- Une note de TP.
- Une note théorique (un contrôle de 1h après tous les TD et TPs).

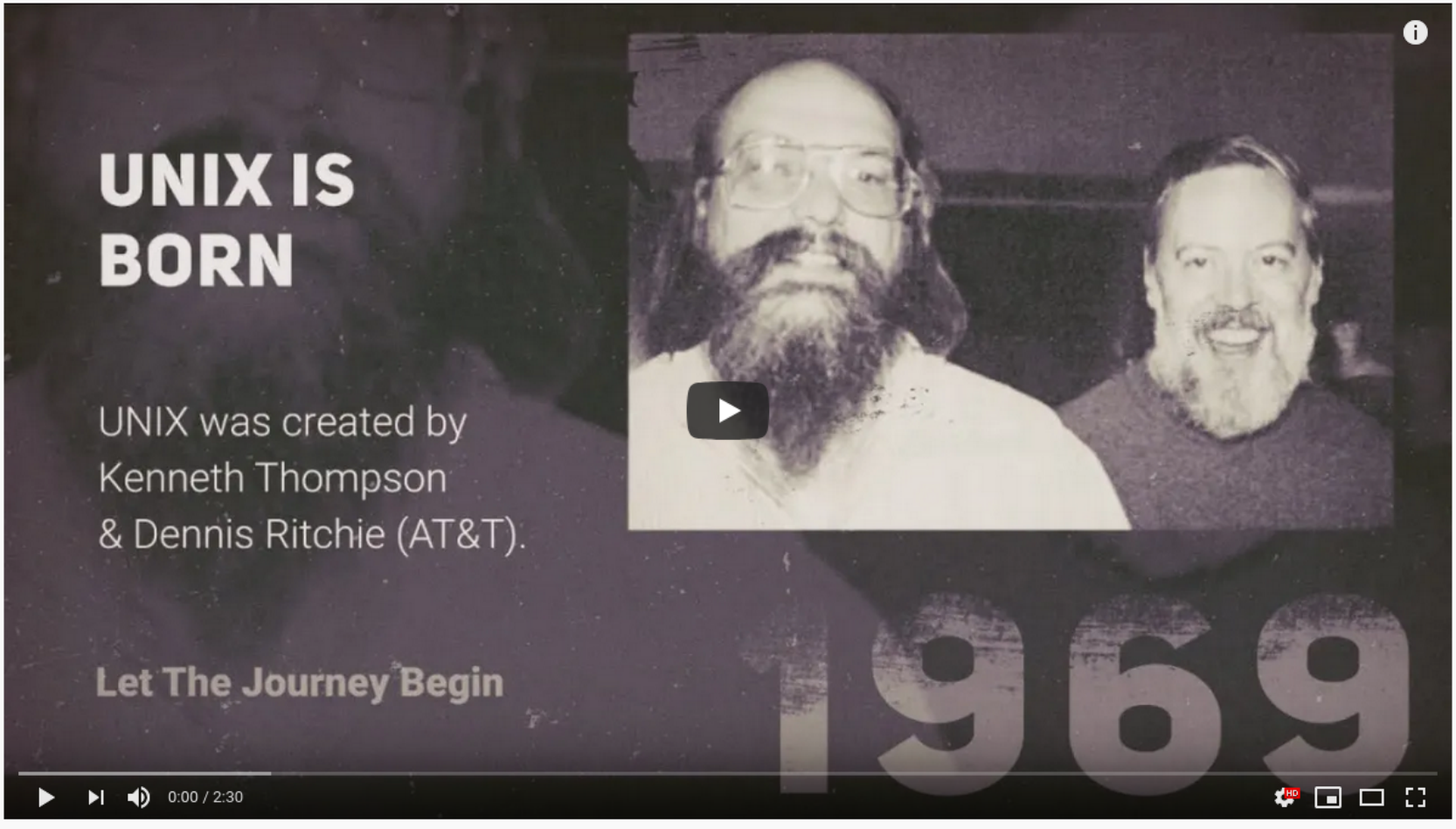
Matériels de cours

https://kirgizov.link/teaching/esirem/unix-and-c/

Aujourd'hui

- Notes historiques
- La documentation Unix
- Littérature
- Philosophie d'Unix
- Architecture
- Arborescence des fichiers
- Écosystème du shell
  - Variable d'environnement PATH
  - Commandes et leurs paramètres
  - Lecture de fichiers
  - Connexion avec le monde extérieur
  - Recherche
  - Modifications
  - Autres commandes
- Flux des données
- Permissions Unix

Notes historiques



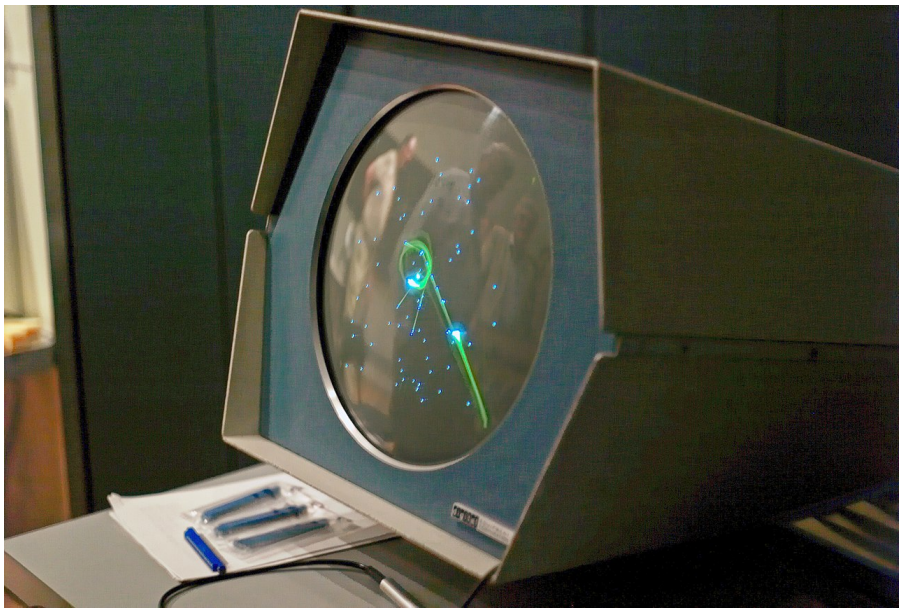
The Great History of UNIX (1969-1999) | 30 Years of UNIX History | UNIX and Linux Forums

https://www.youtube.com/watch?v=DXh2_CTJW9w

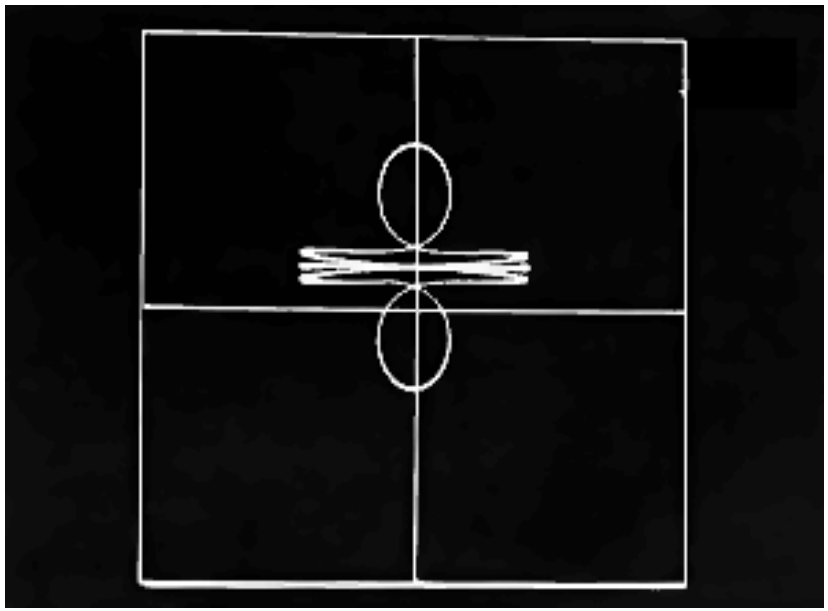
La machine PDP-7.

Les jeux video de l'époque

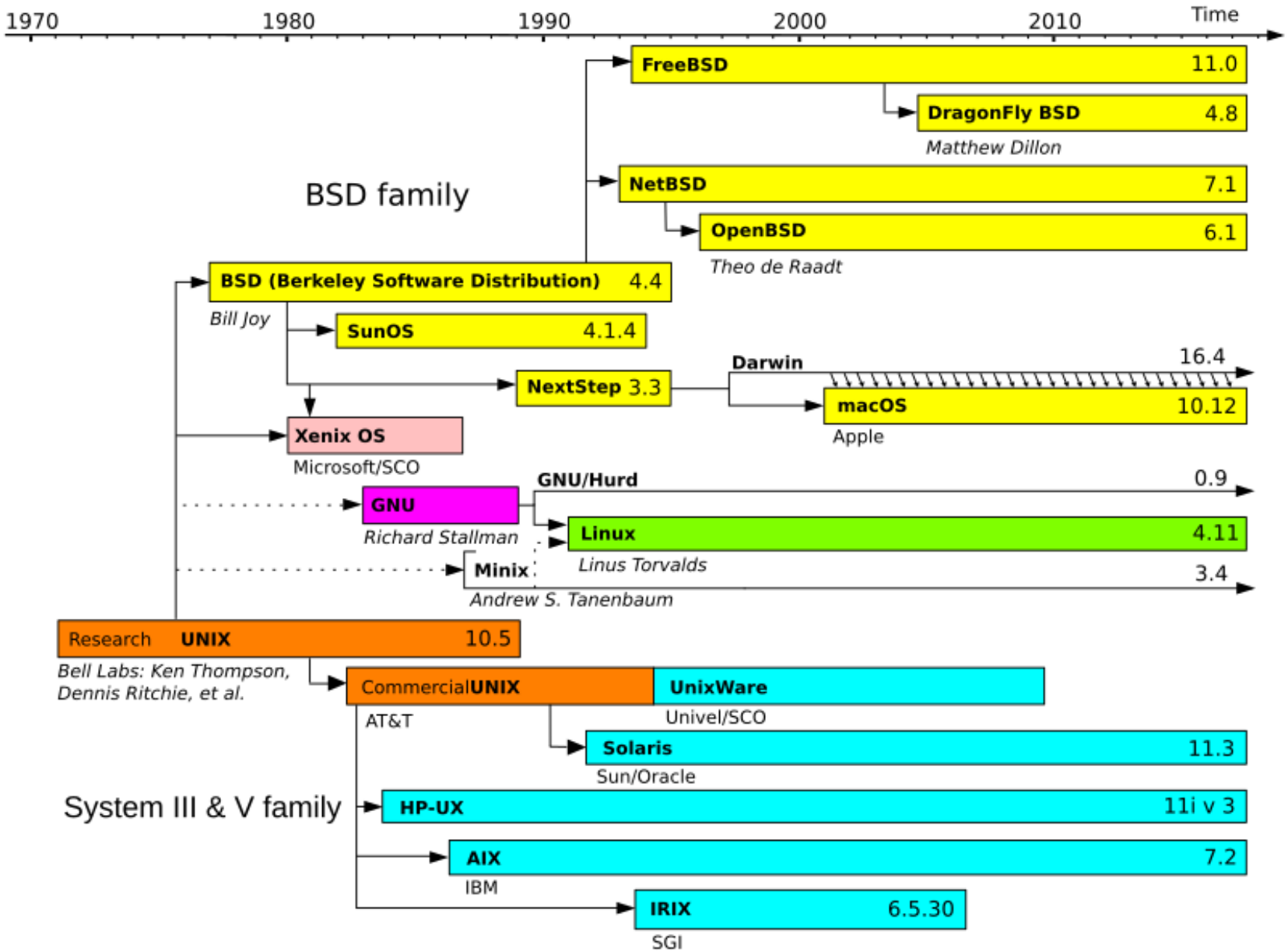
Spacewar !



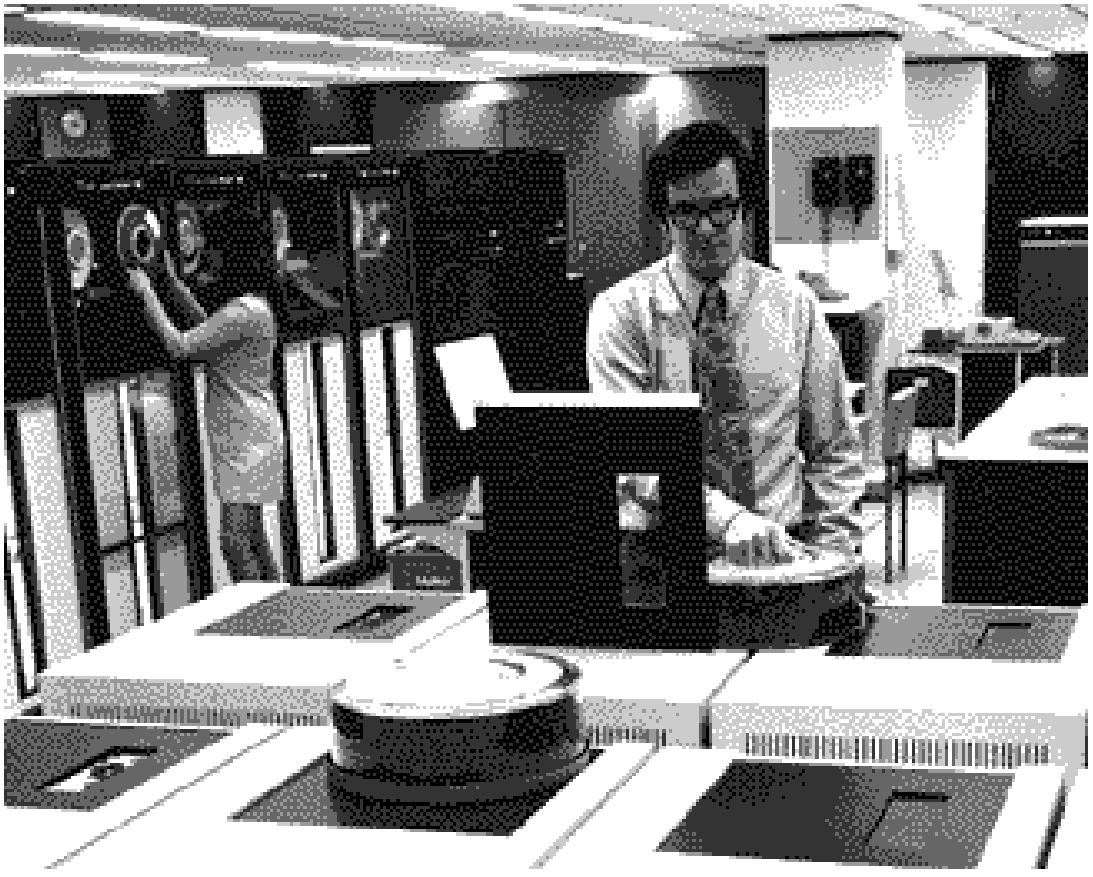
Space Travel



Unix Time Line



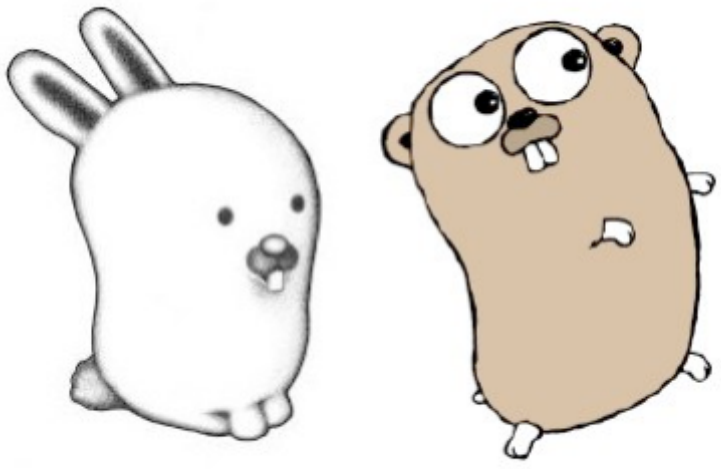
Multics : prédécesseur de Unix



More info `https://www.multicians.org/`

L'avenir d'Unix ?

Deux exemples : la recherche et la pratique.



Un système d'exploitation *Plan 9* et un langage de programation *Go* sont faites par Ken Thompson avec ces collègues : Rob Pike, Robert Griesemer, etc.

UTF-8 a été inventé par Ken Thompson pour Plan 9 en 1992.

Mascots dessinée par Renée French.

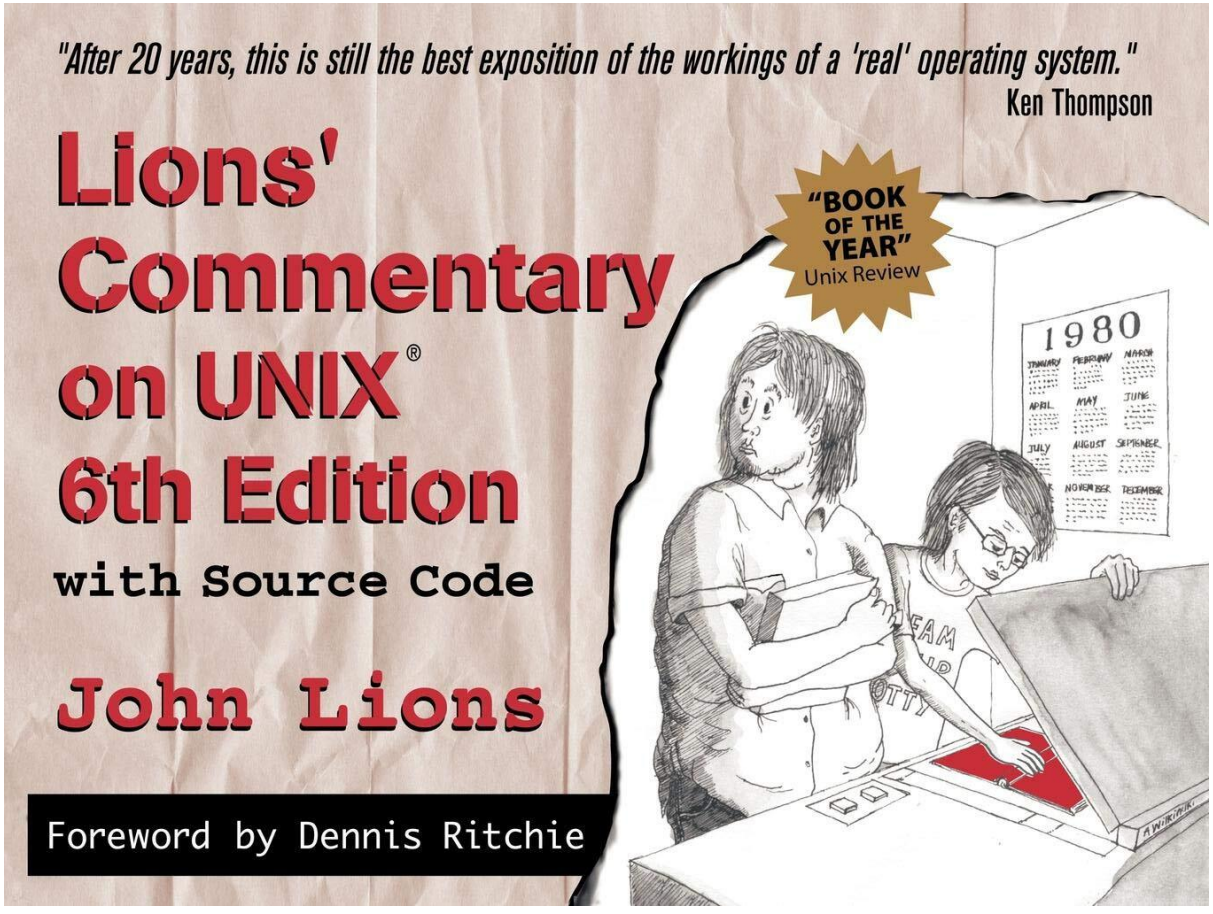
La documentation Unix

La documentation Unix

- Sur le net : tutoriels, forum de discussion, moteur de recherches, RFC, wiki, livres
- Documentation incluse dans le système : `man`, `info`, `help`

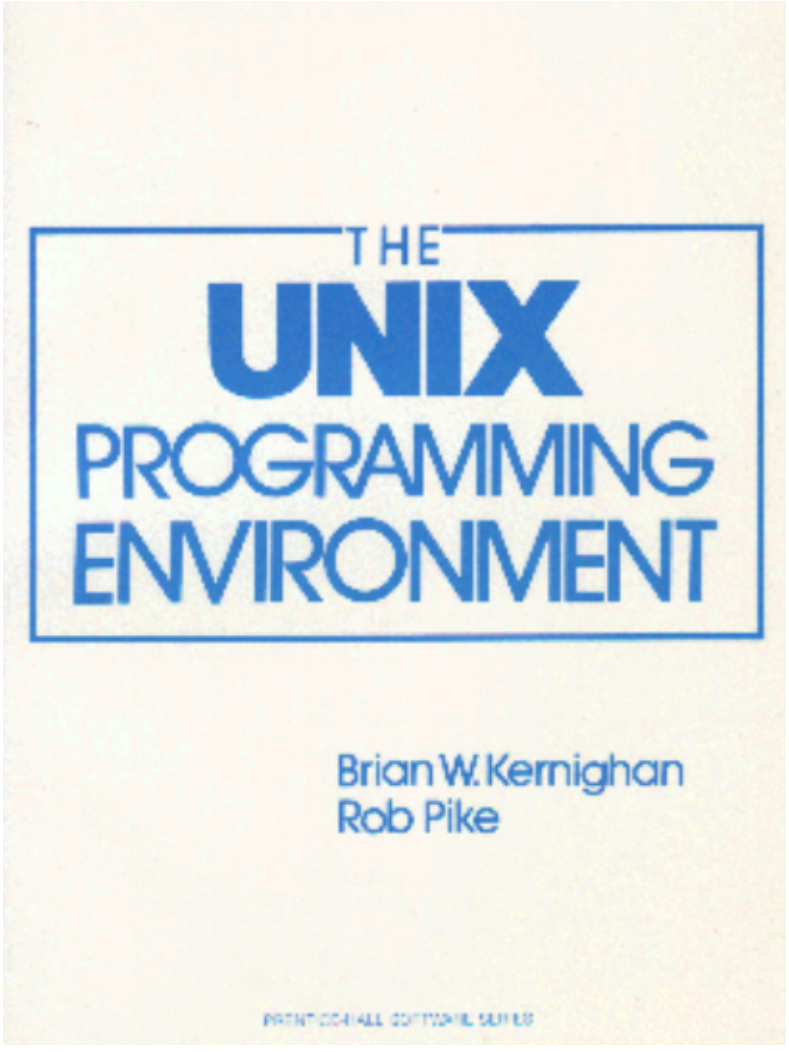
Littérature

Premier livre sur Unix, 1976



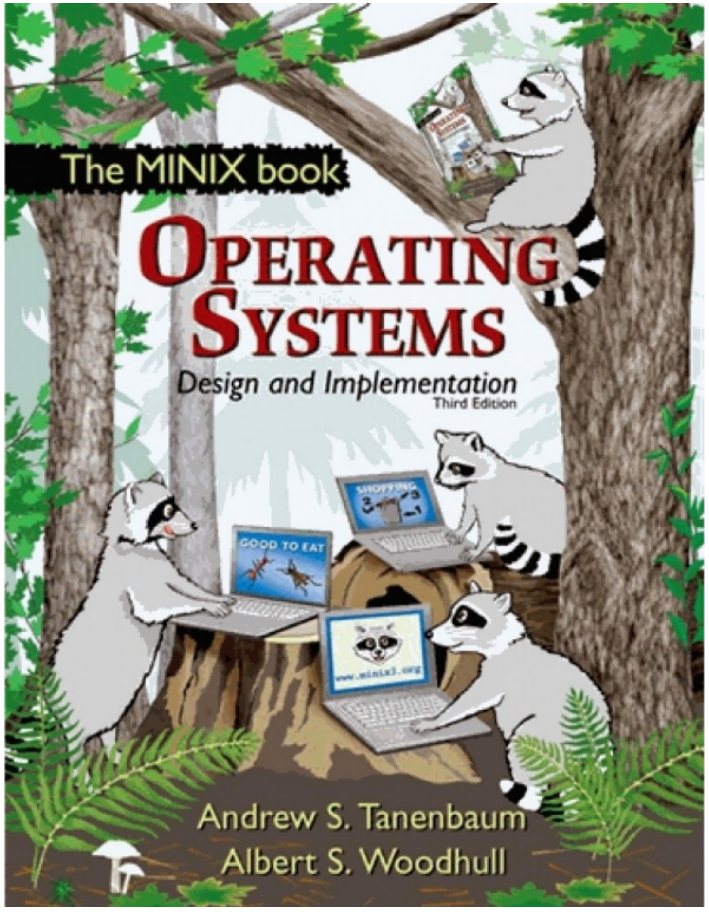
John Lions

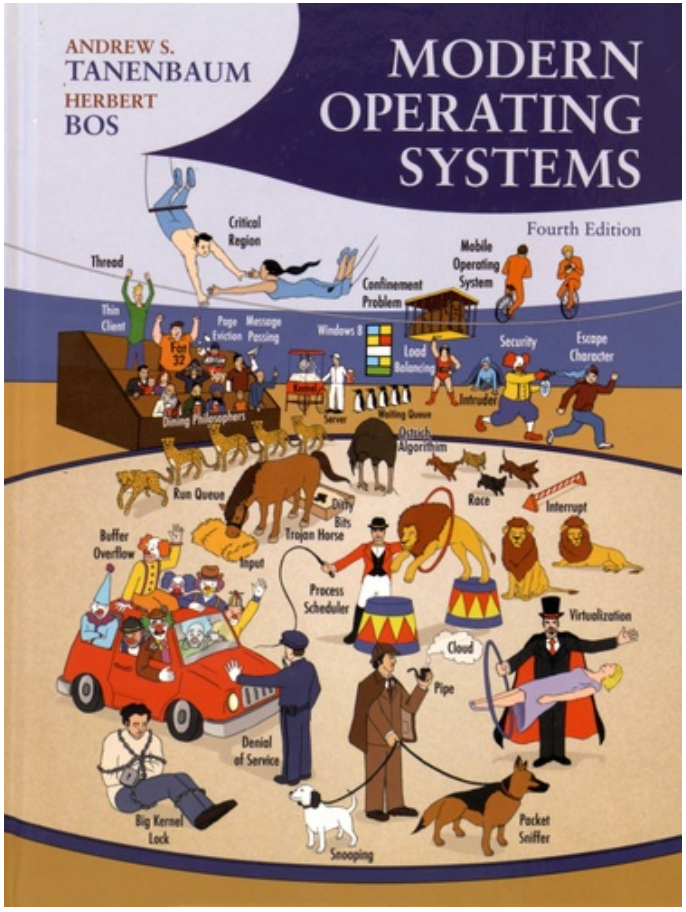
Un autre bon vieux livre



The Unix Programming Environment
Brian W. Kernighan and Rob Pike

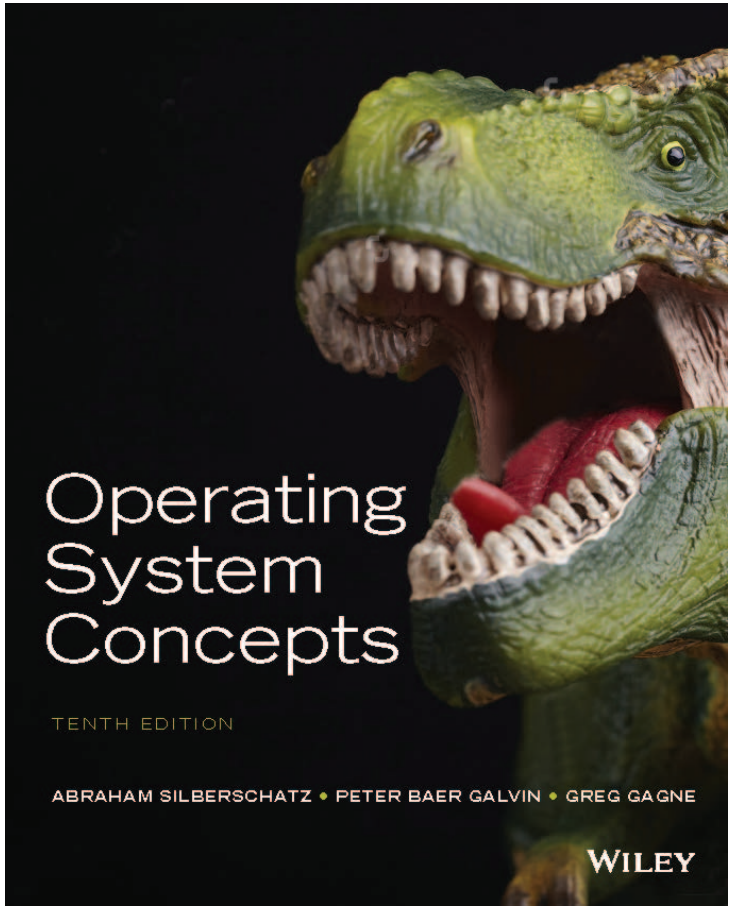
MINIX, papa de Linux





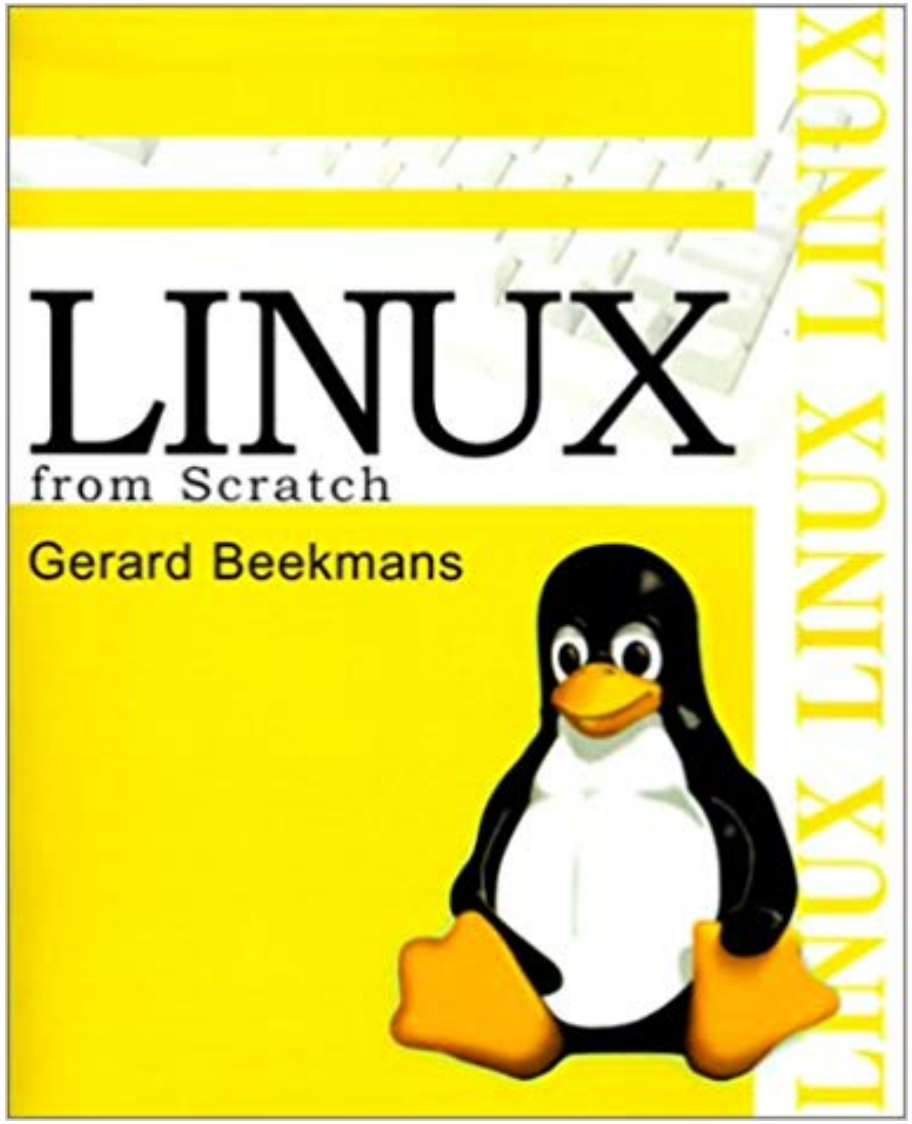
- Operating Systems : Design and Implementation.
  Andrew S. Tanenbaum et Albert S. Woodhull.
- Modern Operating Systems.
  Andrew S. Tanenbaum, Herbert Bos

Un autre bon livre sur les systèmes d'exploitation



`https://www.os-book.com/OS10/index.html` Operating systems concepts
Avi Silberschatz, Peter Baer Galvin, Greg Gagne

Do it yourself !



Le projet *Linux From Scratch* est un livre relatant les diverses étapes pour construire à partir des codes sources un système GNU/Linux.
Par Gerard Beekmans et al.

Avant-garde ? !

`http://9front.org`

Groupes de discussion existent sur IRC et Discord.

Philosophie d'Unix

Philosophie d'Unix

Douglas McIlroy

"Voici la philosophie d'Unix :

- Écrivez des programmes qui effectuent une seule chose et qui le font bien.
- Écrivez des programmes qui collaborent.
- Écrivez des programmes pour gérer des flux de texte, car c'est une interface universelle"

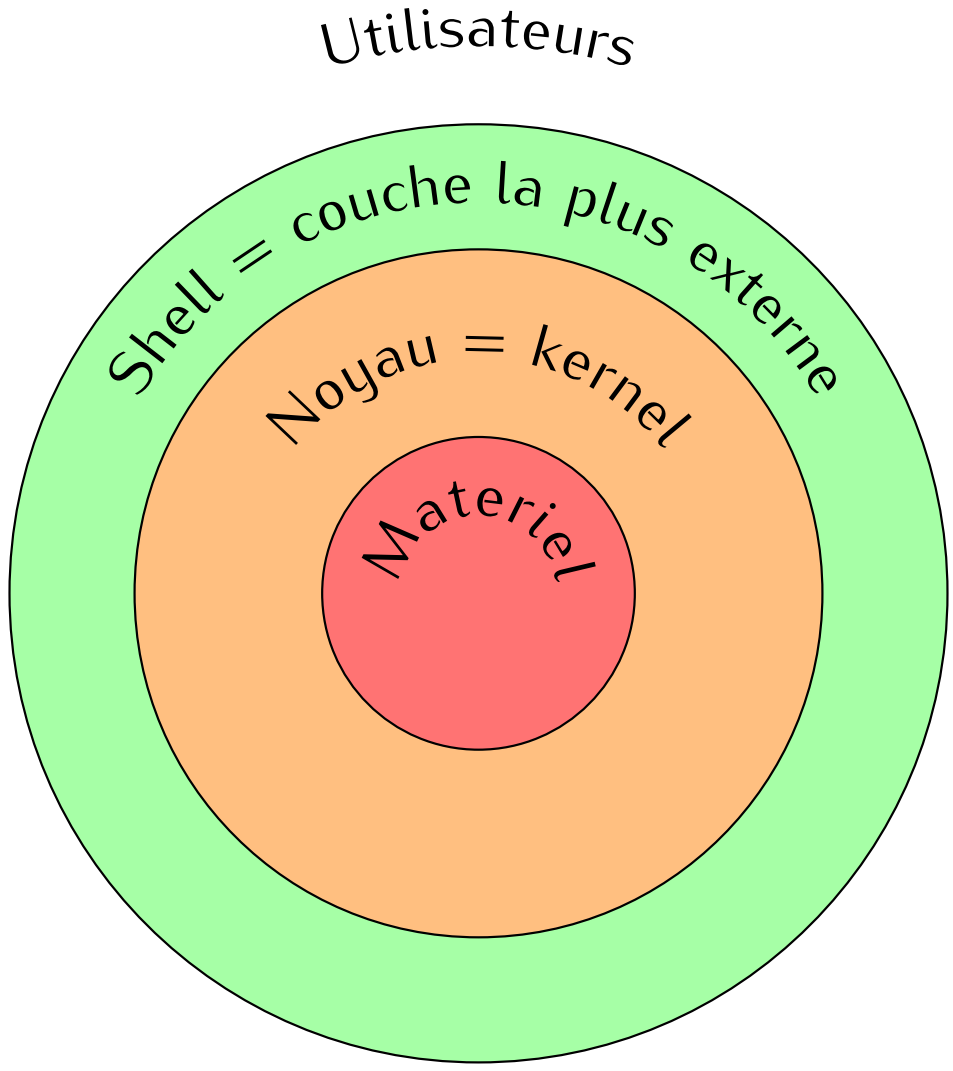
Il y une page sur wikipedia ! `https://en.wikipedia.org/wiki/Unix_philosophy`

Everything is a file

"I think the major good idea in Unix was its clean and simple interface : open, close, read, and write."

– Ken Thompson

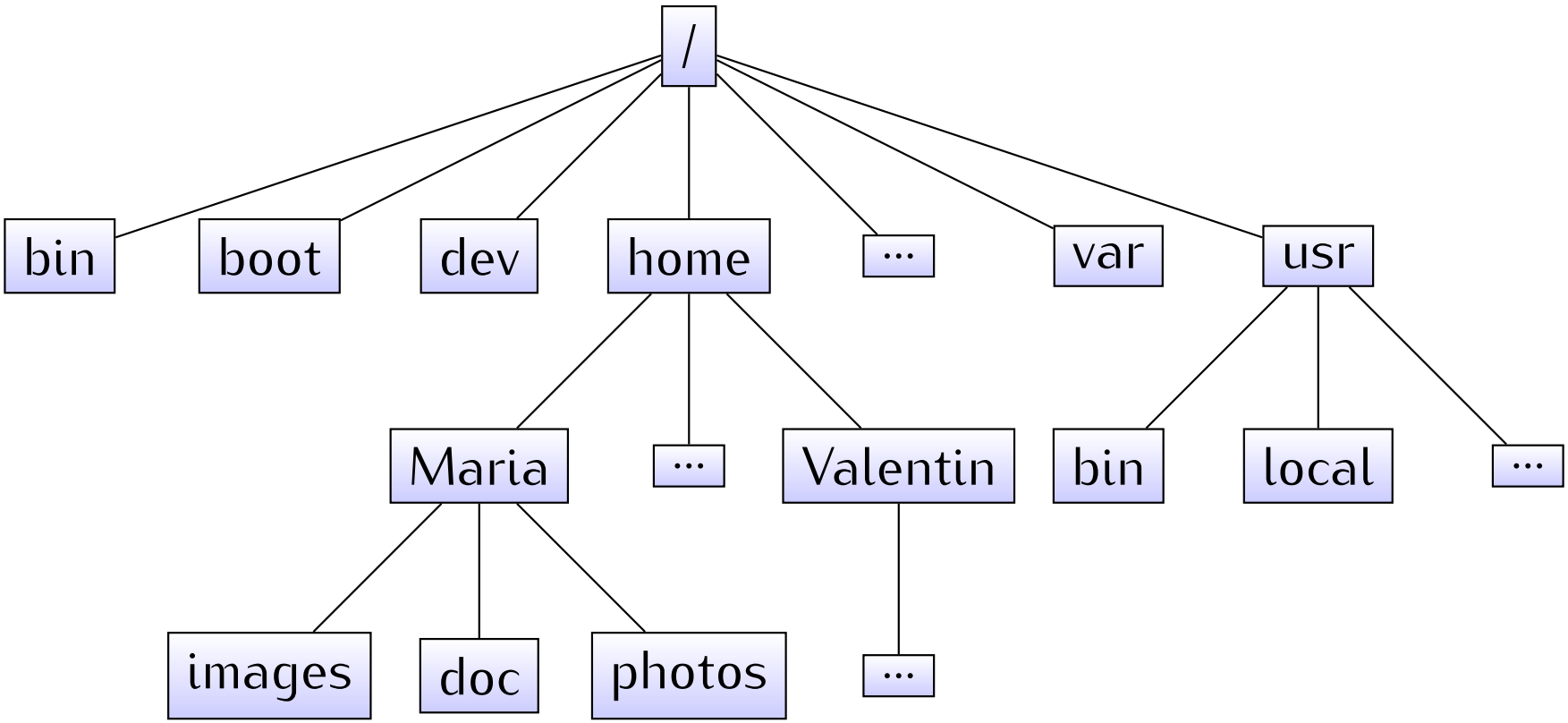
Architecture Unix


Utilisateurs
Shell = couche la plus externe
Noyau = kernel
Materiel

Arborescence des fichiers

Organisation des fichiers

Les fichiers sont organisés en arborescence, dont la racine est /
La documentation : `man hier`



Un chemin est une séquence des noms de noms séparés par une barre oblique (slash, /). Un chemin peut mener vers un fichier, un répertoire ou même nulle part.

Organisation des fichiers. Chemins

Deux répertoire speciales :

- . – rester dans le répertoire courant
- .. – remonter au répertoire parent

Organisation des fichiers. Chemins

Deux répertoire speciales :

- . – rester dans le répertoire courant
- .. – remonter au répertoire parent
- Chemins absolu, ils commencent à la racine.
  - `/repertoire1/repertoire2/`

Organisation des fichiers. Chemins

Deux répertoire speciales :

- . – rester dans le répertoire courant
- .. – remonter au répertoire parent
- Chemins absolu, ils commencent à la racine.
  - `/repertoire1/repertoire2/`
  - `/repertoire1/repertoire2/repertoire3/`

Organisation des fichiers. Chemins

Deux répertoire speciales :

- . – rester dans le répertoire courant
- .. – remonter au répertoire parent
- Chemins absolu, ils commencent à la racine.
  - `/repertoire1/repertoire2/`
  - `/repertoire1/repertoire2/repertoire3/`
  - `/repertoire1/repertoire2/repertoire3/file.txt`

Organisation des fichiers. Chemins

Deux répertoire speciales :

- `.` – rester dans le répertoire courant
- `..` – remonter au répertoire parent
- Chemins absolu, ils commencent à la racine.
  - `/repertoire1/repertoire2/`
  - `/repertoire1/repertoire2/repertoire3/`
  - `/repertoire1/repertoire2/repertoire3/file.txt`
  - `/repertoire1/repertoire2/../file.txt == /repertoire1/file.txt`

Organisation des fichiers. Chemins

Deux répertoire speciales :

- `.` – rester dans le répertoire courant
- `..` – remonter au répertoire parent
- Chemins absolu, ils commencent à la racine.
  - `/repertoire1/repertoire2/`
  - `/repertoire1/repertoire2/repertoire3/`
  - `/repertoire1/repertoire2/repertoire3/file.txt`
  - `/repertoire1/repertoire2/../file.txt == /repertoire1/file.txt`
  - `/repertoire1/repertoire2/../../file.txt == /file.txt`

Organisation des fichiers. Chemins

Deux répertoire speciales :

- `.` – rester dans le répertoire courant
- `..` – remonter au répertoire parent
- Chemins absolu, ils commencent à la racine.
  - `/repertoire1/repertoire2/`
  - `/repertoire1/repertoire2/repertoire3/`
  - `/repertoire1/repertoire2/repertoire3/file.txt`
  - `/repertoire1/repertoire2/../file.txt == /repertoire1/file.txt`
  - `/repertoire1/repertoire2/../../file.txt == /file.txt`
- Chemins relatifs, ils partent du répertoire courant.
  Si le répertoire courant est '`/home/user/truc/`' alors
  - `help/file.txt == /home/user/truc/help/file.txt`
  - `./help/file.txt == /home/user/truc/help/file.txt`
  - `../../help/file.txt == /home/help/file.txt`

Organisation des fichiers. Chemins

Deux répertoire speciales :

- `.` – rester dans le répertoire courant
- `..` – remonter au répertoire parent
- Chemins absolu, ils commencent à la racine.
  - `/repertoire1/repertoire2/`
  - `/repertoire1/repertoire2/repertoire3/`
  - `/repertoire1/repertoire2/repertoire3/file.txt`
  - `/repertoire1/repertoire2/../file.txt == /repertoire1/file.txt`
  - `/repertoire1/repertoire2/../../file.txt == /file.txt`
- Chemins relatifs, ils partent du répertoire courant.
  Si le répertoire courant est '`/home/user/truc/`' alors
  - `help/file.txt == /home/user/truc/help/file.txt`
  - `./help/file.txt == /home/user/truc/help/file.txt`
  - `../../help/file.txt == /home/help/file.txt`

Petite précision :

Les fichiers commençant par un `.` sont des fichiers “cachés”.

Organisation des fichiers. Les commandes essentielles

- `ls` : afficher le contenu d'un répertoire

  Exemple : `ls`
- `cd` : se déplacer dans l'arborescence

  Exemple : `cd /home/user/doc`
- `mv` : déplacer ou renommer

  Exemple : `mv ../file ./repertoire/newfile`
- `cp` : copier des fichiers ou des répertoires

  Exemple : `cp ../file ./repertoire/newfile`
- `rm` : supprimer des fichiers

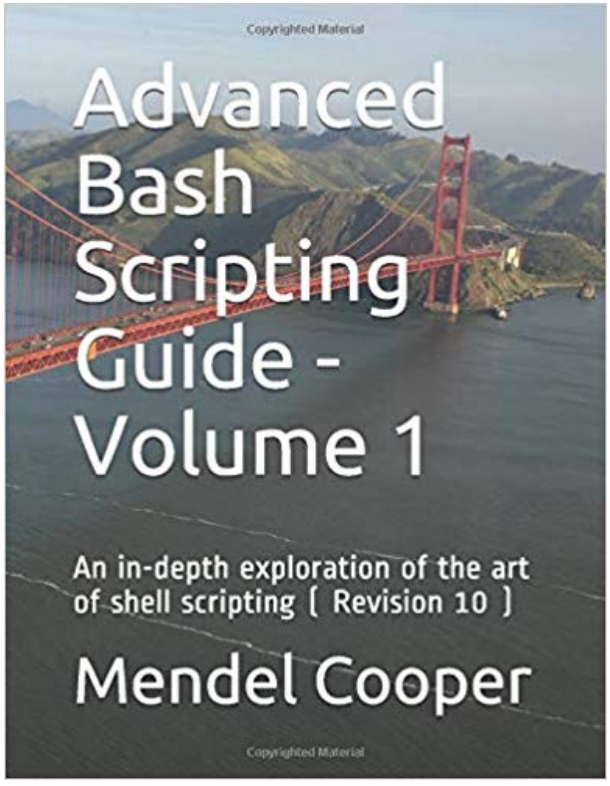
  Exemple : `rm file.txt`
- `mkdir` : créer un dossier

  Exemple : `mkdir /home/egor-letov`
- `rmdir` : supprimer un dossier vide

  Exemple : `rmdir /home/troll`

Écosystème du shell

Littérature : shell



- Advanced Bash-Scripting Guide par Mendel Cooper
  `https://www.tldp.org/LDP/abs/html/`

Littérature : unix man

Unix & Linux Commands Man Page Set	Commands
BSD 2.11	1,352
CentOS 7.0	38,339
Debian 7.7	81,313
FreeBSD 11.0	9,637
HP-UX 10.31 (mods only)	7,209
Linux 2.6	7,299
Minix 2.0	365
NetBSD 6.1.5	7,937
OSF1 5.1 (alpha)	6,070
OSX 10.14 Mojave	13,507
OSX 10.6.2	30,972
OpenDarwin 7.2.1	1,893
OpenSolaris 2009.06	16,086
POSIX 1003.1	3,514
Plan 9	380

`https://www.unix.com/man-page-repository.php`

Il y des mans en français, par exemple :

`http://jp.barralis.com/linux-man/`

Exemple de man

MAN(1) BSD General Commands Manual MAN(1)

NAME

man — display online manual documentation pages

SYNOPSIS

man [**-adho**] [**-t** | **-w**] [**-M** *manpath*] [**-P** *pager*] [**-S** *mansect*] [**-m** *arch*[:*machine*]]
[**-p** [*eprtv*]] [*mansect*] *page* . . .
man **-f** *keyword* . . .
man **-k** *keyword* . . .

DESCRIPTION

The **man** utility finds and displays online manual documentation pages. If *mansect* is provided, **man** restricts the search to the specific section of the manual.

The sections of the manual are:

1. FreeBSD General Commands Manual
2. FreeBSD System Calls Manual
3. FreeBSD Library Functions Manual
4. FreeBSD Kernel Interfaces Manual
5. FreeBSD File Formats Manual
6. FreeBSD Games Manual
7. FreeBSD Miscellaneous Information Manual
8. FreeBSD System Manager's Manual
9. FreeBSD Kernel Developer's Manual

Options that **man** understands:

-M *manpath*

Forces a specific colon separated manual path instead of the default search path. See manpath(1). Overrides the MANPATH environment variable.

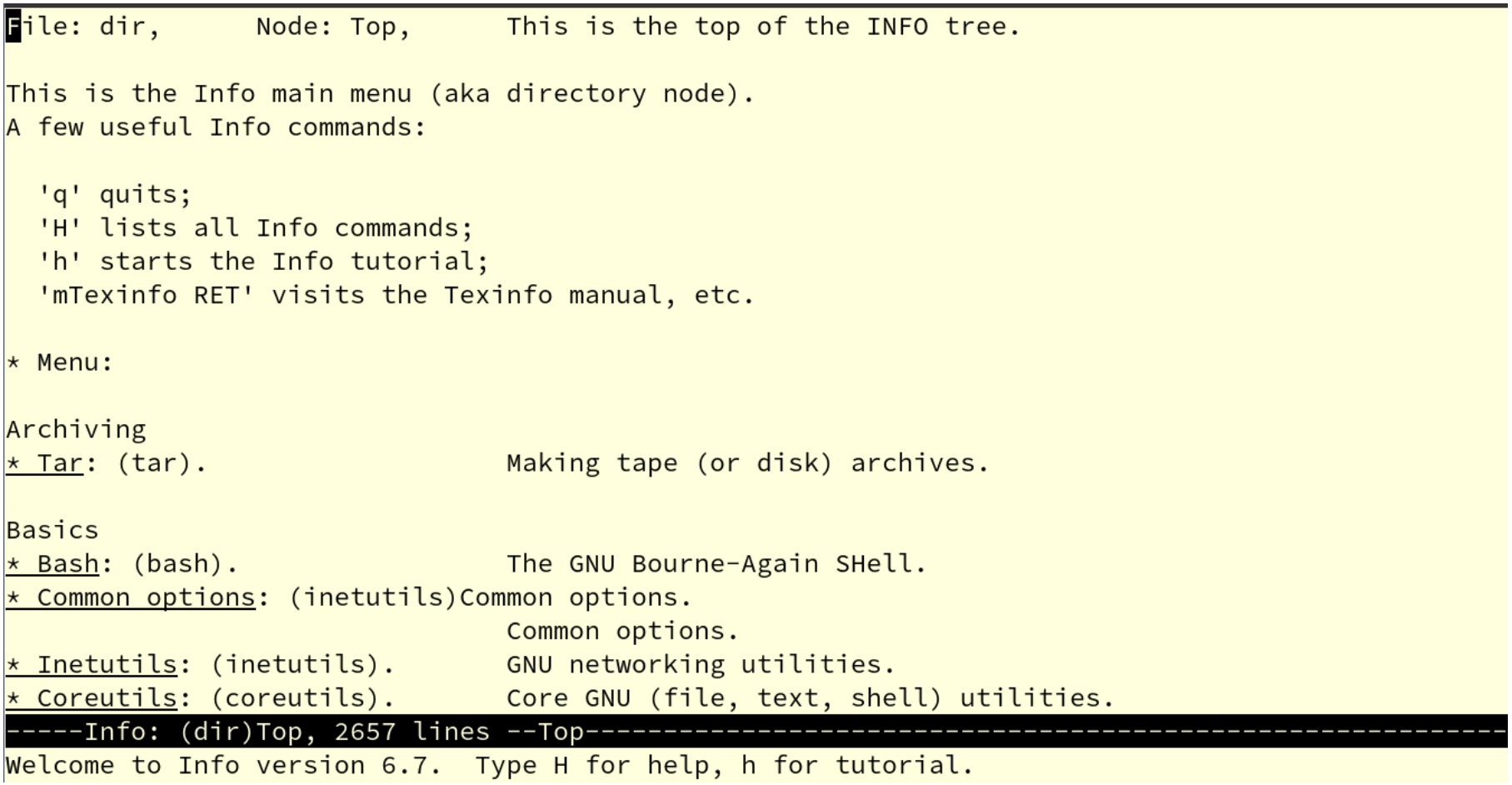
Littérature : info

man
C'est une petite brochure : bref, pratique, et très utile.

info
C'est un livre détaillant les commandes, leurs options et les concepts qui les accompagnent.

Comparer par exemple `man ln` et `info ln`.

info

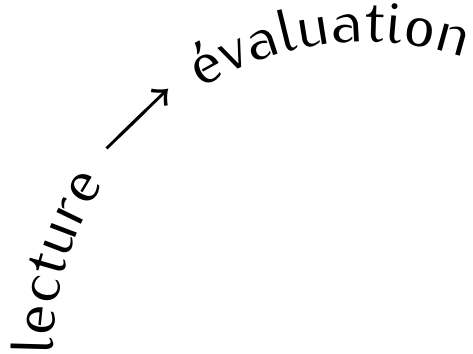

File: dir, Node: Top, This is the top of the INFO tree.
This is the Info main menu (aka directory node).
A few useful Info commands:
'q' quits;
'H' lists all Info commands;
'h' starts the Info tutorial;
'mTexinfo RET' visits the Texinfo manual, etc.
* Menu:
Archiving
* Tar: (tar). Making tape (or disk) archives.
Basics
* Bash: (bash). The GNU Bourne-Again SHell.
* Common options: (inetutils)Common options.
Common options.
* Inetutils: (inetutils). GNU networking utilities.
* Coreutils: (coreutils). Core GNU (file, text, shell) utilities.
-----Info: (dir)Top, 2657 lines --Top-----
Welcome to Info version 6.7. Type H for help, h for tutorial.

REPL : read–eval–print loop

Shell d'Unix vous permet de communiquer avec le système d'exploitation en boucle :

REPL : read–eval–print loop

Shell d'Unix vous permet de communiquer avec le système d'exploitation en boucle :

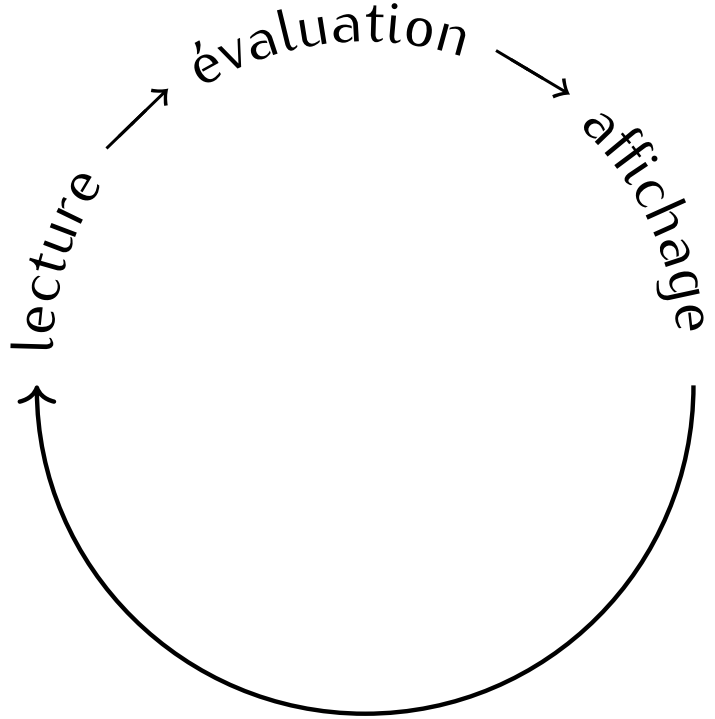


REPL : read–eval–print loop

Shell d'Unix vous permet de communiquer avec le système d'exploitation en boucle :

REPL : read–eval–print loop

Shell d'Unix vous permet de communiquer avec le système d'exploitation en boucle :



Où se trouvent les commandes unix ?

Où se trouvent les commandes unix ?

Dans des répertoires dont les noms sont contenus dans la variable d'environnement `$PATH`

Où se trouvent les commandes unix ?

Dans des répertoires dont les noms sont contenus dans la variable d'environnement `$PATH`

Exemple :

```
echo $PATH
/usr/local/bin:/usr/local/sbin:/usr/bin:/usr/bin/
```

Commandes et leurs paramètres

Commandes et leurs paramètres

Les *paramètres* sont séparés par des espaces, si vous voulez qu'un paramètre contient un espace, vous devez soit y échapper par un antislash "\", soit utiliser des apostrophes ou des guillemets.

Commandes et leurs paramètres

Les *paramètres* sont séparés par des espaces, si vous voulez qu'un paramètre contient un espace, vous devez soit y échapper par un antislash "\", soit utiliser des apostrophes ou des guillemets.

Les *options* sont des paramètres commençant par le signe moins "-".

- Les *options courtes* ont la forme de "-x" où "x" est un seul caractère.
- Les *options longues* ont la forme de "--bla-bla-bla".

Commandes et leurs paramètres

Les *paramètres* sont séparés par des espaces, si vous voulez qu'un paramètre contient un espace, vous devez soit y échapper par un antislash "\", soit utiliser des apostrophes ou des guillemets.

Les *options* sont des paramètres commençant par le signe moins "-".

- Les *options courtes* ont la forme de "-x" où "x" est un seul caractère.
- Les *options longues* ont la forme de "--bla-bla-bla".

Généralement, l'ordre des options n'a pas d'importance.

Commandes et leurs paramètres

Les *paramètres* sont séparés par des espaces, si vous voulez qu'un paramètre contient un espace, vous devez soit y échapper par un antislash "\", soit utiliser des apostrophes ou des guillemets.

Les *options* sont des paramètres commençant par le signe moins "-".

- Les *options courtes* ont la forme de "-x" où "x" est un seul caractère.
- Les *options longues* ont la forme de "--bla-bla-bla".

Généralement, l'ordre des options n'a pas d'importance.
Les options courtes peuvent être regroupées, par exemple, les trois commandes suivantes sont équivalentes

```
ls -l -a
ls -la
ls -al
```

Commandes et leurs paramètres

Exemple

```
boulangerie -a -b --foo-bar fichier1.txt "tram param"
```

- $0 : nom de la commande, 'boulangerie'
- $1 : une option courte, '-a'
- $2 : une autre option courte, '-b'
- $3 : une longe option '--foo-bar'
- $4 : un parameter, 'fichier1.txt'
- $5 : un autre parameter, 'tram param'

Lecture de fichiers

Lecture : cat

```
CAT(1)                                                          User Commands
                                        CAT(1)

NAME
       cat - concatenate files and print on the standard output

SYNOPSIS
       cat [OPTION]... [FILE]...

DESCRIPTION
       Concatenate FILE(s) to standard output.

       With no FILE, or when FILE is -, read standard input.

       -A, --show-all
              equivalent to -vET

       -b, --number-nonblank
              number nonempty output lines, overrides -n

       -e     equivalent to -vE

       -E, --show-ends
              display $ at end of each line

       -n, --number
              number all output lines
 Manual page cat(1) line 1/69 36% (press h for help or q to quit)
```

Lecture de premières lignes : head

```
HEAD(1)             User Commands             HEAD(1)

NAME
       head - output the first part of files

SYNOPSIS
       head [OPTION]... [FILE]...

DESCRIPTION
       Print the first 10 lines of each FILE to stan-
       dard output.  With more than one FILE, precede
       each with a header giving the file name.

       With no FILE, or when FILE is -, read standard
       input.

       Mandatory arguments to long options are manda-
       tory for short options too.

       -c, --bytes=[-]NUM
              print the first NUM bytes of each file;
              with the leading '-', print all but the
              last NUM bytes of each file

       -n, --lines=[-]NUM
              print  the  first  NUM lines instead of
              the first 10;  with  the  leading  '-',
              print  all  but  the  last NUM lines of
              each file

       -q, --quiet, --silent
              never print headers giving file names
```

Lecture de dernières lignes : tail

```
TAIL(1)                    User Commands                    TAIL(1)

NAME
       tail - output the last part of files

SYNOPSIS
       tail [OPTION]... [FILE]...

DESCRIPTION
       Print  the last 10 lines of each FILE to standard output.
       With more than one FILE, precede each with a header  giv-
       ing the file name.

       With no FILE, or when FILE is -, read standard input.

       Mandatory  arguments  to  long  options are mandatory for
       short options too.

       -c, --bytes=[+]NUM
              output the last NUM bytes; or use -c +NUM to  out-
              put starting with byte NUM of each file

       -f, --follow[={name|descriptor}]
              output appended data as the file grows;

              an absent option argument means 'descriptor'

       -F     same as --follow=name --retry

       -n, --lines=[+]NUM
              output the last NUM lines, instead of the last 10;
              or use -n +NUM to output starting with line NUM
```

Lecture inversée : tac

```
TAC(1)                          User Commands                          TAC(1)

NAME
       tac - concatenate and print files in reverse

SYNOPSIS
       tac [OPTION]... [FILE]...

DESCRIPTION
       Write each FILE to standard output, last line first.

       With no FILE, or when FILE is -, read standard input.

       Mandatory  arguments  to  long  options are mandatory for
       short options too.

       -b, --before
              attach the separator before instead of after

       -r, --regex
              interpret the separator as a regular expression

       -s, --separator=STRING
              use STRING as the separator instead of newline
```

wc : word count

Compter les lignes dans un fichier.

```
wc -l fichier
```

Compter les octets

```
wc -c fichier
```

Compter les mots

```
wc -w fichier
```

Affichage de gros fichiers

- **more** : affichage page par page
- **less** : affichage page par page et navigation

Connexion avec le monde extérieur

Les données publiques

https://www.data.gouv.fr/fr/datasets/ficher-des-prenoms-de-1900-a-2018/

Communication avec les serveurs

Exemple, téléchargement :

```
wget  https://www.insee.fr/fr/statistiques/fichier/2540004/nat2018_csv.zip
```

Autres commandes existent : `curl`, `telnet`, `netcat`, `ping`, `etc...`

Recherche

grep : rechercher des lignes contenant le motif spécifié

La commande **grep** permet de filtrer les lignes d'un fichier et de n'afficher que certaines d'entre elles.

Utile pour savoir si un fichier contient un mot donné. Des nombreuses autres applications existent.

Usage habituelle

```
grep motif fichier
```

grep : rechercher des lignes contenant le motif spécifié

La commande **grep** permet de filtrer les lignes d'un fichier et de n'afficher que certaines d'entre elles.

Utile pour savoir si un fichier contient un mot donné. Des nombreuses autres applications existent.

Usage habituelle

```
grep motif fichier
```

Quelques options

- option -v : afficher les lignes qui ne contiennent pas le motif
- option -n : numéroter les lignes résultat
- option -e : pour préciser plusieurs motifs
- option -E : utiliser les expressions régulières étendues
- ...
- sans fichier en paramètre : utilisation de l'entrée standard

grep, exemple I

```
➜  ~ grep motif /usr/share/dict/words
leitmotif
leitmotif's
leitmotifs
motif
motif's
motifs
➜  ~
```

grep, exemple II

```
➜  ~ grep -n recurs /usr/share/dict/words
83967:precursor
83968:precursor's
83969:precursors
83970:precursory
89256:recurs
89257:recursion
89258:recursions
89259:recursive
89260:recursively
```

grep, exemple III

```
➜  ~ grep -e dada -e cubis /usr/share/dict/words
cubism
cubism's
cubist
cubist's
cubists
dadaism
dadaism's
dadaist
dadaist's
dadaists
➜  ~
```

Modifications

Éditeurs de text

- nano
- vi
- emacs
- ainsi que d'autres : ed, acme, sam, gedit, kate, joe, ...

sort : trier les lignes d'un fichier texte

Exemple 1 : trier dans l'ordre lexicographique croissant

```
sort fichier.txt
```

sort : trier les lignes d'un fichier texte

Exemple 1 : trier dans l'ordre lexicographique croissant

```
sort fichier.txt
```

Exemple 2 : trier dans l'ordre lexicographique décroissant

```
sort -r fichier.csv
```

sort : trier les lignes d'un fichier texte

Exemple 1 : trier dans l'ordre lexicographique croissant

```
sort fichier.txt
```

Exemple 2 : trier dans l'ordre lexicographique décroissant

```
sort -r fichier.csv
```

Exemple 3 : trier dans l'ordre numérique croissant

```
sort -n fichier.html
```

Exemple 4 : trier dans l'ordre numérique décroissant

```
sort -r -n fichier.xml
```

ou bien

```
sort --reverse --numeric-sort fichier.xml
sort -rn fichier.xml
sort -nr fichier.xml
```

uniq : supprimer les mêmes lignes qui se succèdent

fichier.txt :

```
foo
foo
bar
baz
baz
foo
```

uniq : supprimer les mêmes lignes qui se succèdent

fichier.txt :

```
foo
foo
bar
baz
baz
foo
```

Exemple 1 : `uniq fichier.txt`

```
foo
bar
baz
foo
```

uniq : supprimer les mêmes lignes qui se succèdent

fichier.txt :

```
foo
foo
bar
baz
baz
foo
```

Exemple 1 : `uniq fichier.txt`

```
foo
bar
baz
foo
```

Exemple 2 : `uniq -c fichier.txt`

```
2 foo
1 bar
2 baz
1 foo
```

Autres commandes

Texte : cut, paste, join, tr, sed, awk, diff, patch

Délégation : xargs, GNU Parallel

Réseau : netcat, curl, telnet, ping, tracepath

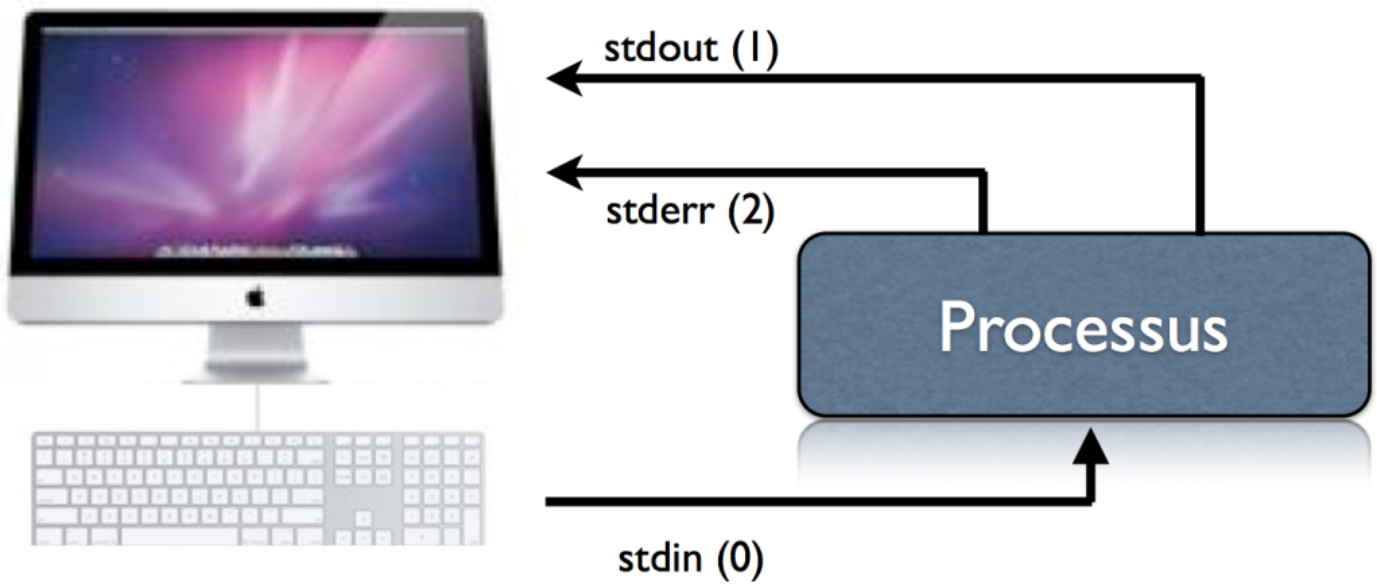
Flux des données

Entrées sortie d'un processus

Un processus

est un programme en cours d'exécution.

- stdin (0) : entrée standard
- stdout (1) : sortie standard
- stderr (2) : sortie erreur standard



L'image vient du cours de Benoît Darties.

`https://benoit.darties.fr/`

Redirections

Utilisation :
`programme (opérateur) fichier`

Opérateurs de redirection vers / depuis un fichier

>	écraser le fichier avec le contenu du stdout
>>	ajouter à la fin du fichier le contenu du stdout
2>	écraser le fichier avec le contenu du stderr
2>>	ajouter à la fin du fichier le contenu du stderr
<	envoyer le contenu du fichier à l'entrée standard

Redirections

Exemple :

```
grep truc fichier.txt > res.pect
```

Signification : enregistrer dans le fichier '`res.pect`' la liste de lignes du fichier '`fichier.txt`' contenant une séquence de caractères '`truc`'

Tubes, pipes, pipelines

Exemple :

```
grep truc fichier.txt | wc -l
```

Signification : compter le nombre de lignes d'un fichier contenant une séquence de caractères 'truc'.

Tubes, pipes, pipelines

Exemple :

```
grep truc fichier.txt | wc -l
```

Signification : compter le nombre de lignes d'un fichier contenant une séquence de caractères 'truc'.

Exemple :

```
head -n 100 fichier.txt | grep truc | wc -l
```

Signification : compter le nombre de lignes parmi 100 premières lignes du fichier contenant une séquence de caractères 'truc'.

Question

Exemple :

```
head -n 100 fichier.txt | grep truc | wc -l > fi
```

Signification : ????

Question

Exemple :

```
head -n 100 fichier.txt | grep truc | wc -l > fi
```

Signification : ????
enregistrer dans le fichier 'fi' le nombre de lignes parmi 100 premières lignes du fichier contenant une séquence de caractères 'truc'.

Tubes, pipes, pipelines

Le concept a été inventé par Malcolm Douglas McIlroy.
La notation “|” a été inventé par Ken Thompson.

`http://cs.bell-labs.co/who/ken/`
`https://www.cs.dartmouth.edu/~doug/`
`https://en.wikipedia.org/wiki/Pipeline_(Unix)`

Permissions Unix

Permissions Unix

- Systèmes Unix sont des systèmes multi-utilisateurs.
- Un utilisateur est membre d'un ou plusieurs groupes.
- Chaque fichier ou répertoire appartient à un utilisateur et à un groupe.
- Trois droits fondamentaux existent
  - lecture (r, read)
  - écriture (w, write)
  - exécution (x, eXecute)
- Le propriétaire, le groupe de propriétaires et tous les autres utilisateurs peuvent avoir des droits différents sur un fichier ou un répertoire.

La documentation

- `man chmod`
- `https://wiki.debian.org/Permissions`
- `info libc "File Attributes"`

Exemple : autoriser l'exécution du fichier '/home/user/file.txt' :

```
chmod +x /home/user/file.txt
```

Questions ?

sergey.kirgizov@u-bourgogne.fr

https://kirgizov.link/teaching/esirem/unix-and-c