Introduction à la Programmation : Environnement de travail

Laboratoire d'Intelligence Artificielle
Faculté I&C

©EPFL 2024
J. Sam

Aspects logiciels d'un ordinateur

Pour fonctionner, un ordinateur doit pouvoir interagir avec l'environnement :

- comprendre, c'est-à-dire ici traiter, les informations lui provenant (clic de souris, touche clavier, ...)
- produire des sorties (sons, image écran, ...)

Cela se fait grâce à des **programmes** (ou **logiciels**) dont le plus fondamental, est le système d'exploitation.
Le système d'exploitation est responsable de la gestion des interactions entre l'unité centrale et ses périphériques,
Exemples : MacOS X, **Linux**, Solaris, Windows...

©EPFL 2024
J. Sam

Catégories de Logiciels

logiciels d'application traitement de tâches spécifiques aux utilisateurs

traitements de textes, tableurs, logiciels de comptabilité, CAO, ...

logiciels utilitaires servant au développement des applications

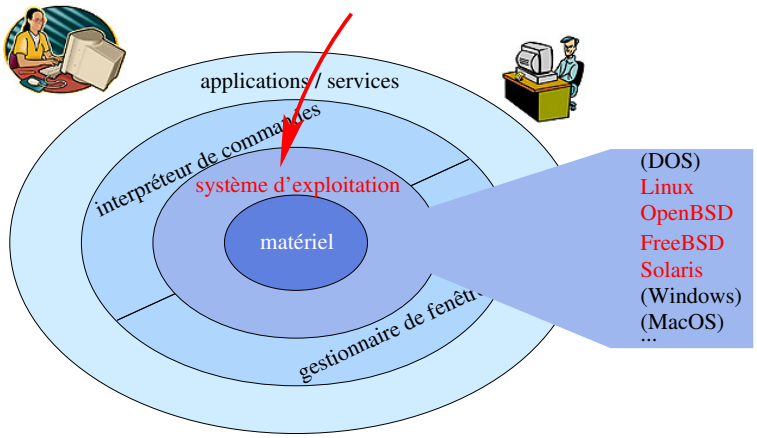
assembleurs, compilateurs, dévermineurs, gestionnaires de versions, gestionnaires de fenêtres, librairies d'outils, ...

logiciels systèmes regroupés dans le **système d'exploitation**

☞ présents au cœur de l'ordinateur, ces logiciels sont à la base de toute exploitation, coordonnant les tâches essentielles à la bonne marche du matériel.

©EPFL 2024
J. Sam

Système d'exploitation

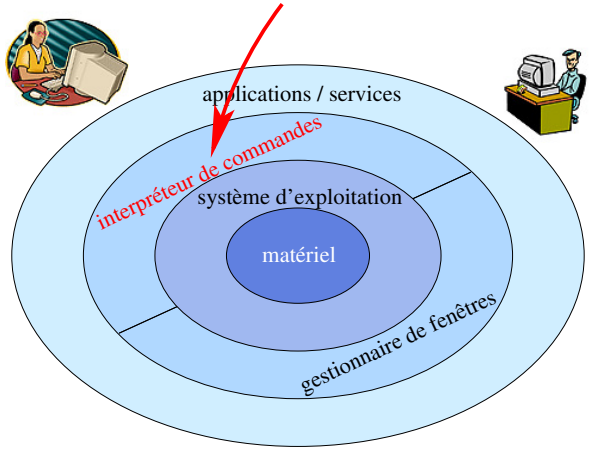


©EPFL 2024
J. Sam

Interaction avec Linux

Comment interagir avec votre système d'exploitation ?

☞ avec un **interpréteur de commande** (**shell**)



Parmi les shells Unix les plus utilisés, citons : Bourne [Again] shell (sh et bash), C shell (csh), Z shell (zsh), et celui présent par défaut sur les comptes du cours, l'Enhanced C shell (tcsh)

©EPFL 2024
J. Sam

Interpréteur de commandes (1)

Pour interagir avec l'utilisateur, un système informatique doit disposer au minimum d'un **interpréteur de commandes** (shell)

Contrairement à d'autres architectures moins modulaires, l'interpréteur de commandes (ainsi que le gestionnaire de fenêtres) des systèmes de type UNIX est un composant externe au SE.
Ne faisant pas directement partie du système, ils peuvent être changés à souhait.

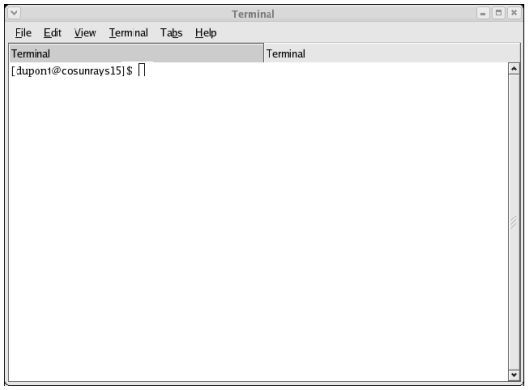
Le shell attend les ordres que l'utilisateur transmet par le biais de l'interface, décode et décompose ces ordres en actions élémentaires, et finalement réalise ces actions en interagissant avec le système d'exploitation.

Parmi les shells Unix les plus utilisés, citons : Bourne [Again] shell (sh et bash), C shell (csh), Z shell (zsh), et celui présent par défaut sur les comptes du cours, l'Enhanced C shell (tcsh).

©EPFL 2024
J. Sam

Interpréteur de commandes (2)

Comment se présentera concrètement votre interpréteur de commande ?
Dans une fenêtre terminal invoquable soit en cliquant sur une option du menu soit par l'usage de la commande `xterm`



©EPFL 2024
J. Sam

Interpréteur de commandes (3)

Depuis un interpréteur de commandes vous aurez la possibilité de :

- lancer des programmes (par exemple un navigateur web, commande `firefox`)
- exécuter des commandes Unix (on peut aussi regrouper plusieurs commandes dans un fichier alors appelé script).
- définir des variables d'environnement,
- renommer ou définir de nouvelles commandes (`alias`), etc...

La plupart des interpréteurs offrent également des facilités d'édition comme le rappel des commandes précédentes (historique des commandes), la complétion (complète le nom du fichier lorsqu'il n'y a plus d'ambiguïté), la correction en cas de commande invalide, ...

©EPFL 2024
J. Sam

La commande `man`

`man` permet d'accèder à l'aide du sytème (« page de `man`uel »)

Utilisations :
`man` *nom*
`man` *section nom*
Exemples : `man tcsh` `man ls` `man man`

Les man-pages sont organisées en différentes **sections** :

1	commandes et programmes	5	formats de fichiers
2	appels systèmes (noyau)	6	jeux
3	bibliothèques logicielles	7	divers
4	fichiers spéciaux (`/dev`)	8	administration système

Comparer :

`man printf`	et	`man 3 printf`
`man time`	et	`man 2 time`
`man man`	et	`man 7 man`

`man -a` *nom* pour avoir **toutes** les man-pages portant sur ce *nom*.
('`q`' pour quitter une manpage et passer à la suivante)

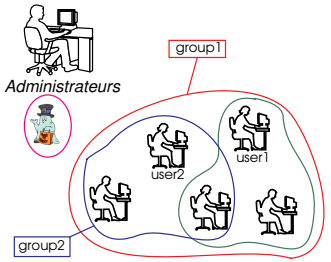
©EPFL 2024
J. Sam
EPFL

Utilisateurs et groupes

Architectures multi-utilisateurs

- identifier les personnes pouvant travailler avec le système
- assurer la confidentialité de leurs données
- parfois, leur facturer les ressources utilisées

☞ 2 entités : les **utilisateurs**, et les **groupes** d'utilisateurs



Les groupes permettent de définir des droits communs à un ensemble d'utilisateurs (quelles ressources utilisables, dans quelles limites, quels droits d'accès, ...)

☞ Chaque utilisateur appartient à au moins 1 groupe
(Essayez la commande `groups`)

©EPFL 2024
J. Sam
EPFL

Quotas

Plusieurs personnes utilisent les mêmes ressources

☞ s'assurer qu'aucun utilisateur ne les monopolise (au détriment des autres)

Par exemple avec les disques durs :
☞ Le **quota disque** représente l'espace maximal qu'un utilisateur peut occuper sur le disque.
Lorsqu'un utilisateur a atteint son quota, le système refusera toute tentative conduisant à une augmentation de l'espace utilisé.
Pour vérifier votre utilisation d'espace disque sur MyNAS (sauvegarde de fichiers au niveau de l'Ecole) :

- utilisez soit l'interface Web : `http://mynas.epfl.ch/`
- soit en utilisant la commande (= dans un terminal) :
  `df -h /myfiles`

©EPFL 2024
J. Sam
EPFL

Système de fichiers

Le concept de **fichiers** est une structure adaptée aux mémoires de masse permettant de regrouper des données.

Un fichier c'est une **collection ordonnée de données**, représentant une entité pour l'utilisateur.

Le **système d'exploitation** va donner corps au concept de fichiers, c'est-à-dire les gérer : les créer, détruire, modifier, lire, et offrir la possibilité de les désigner par des noms.

Dans le cas de systèmes multi-utilisateurs, il faut de plus **assurer la confidentialité** de ces fichiers, en protégant leur contenu du regard des autres utilisateurs.

©EPFL 2024
J. Sam
EPFL

Système de fichiers (2)

Pour assurer la gestion des fichiers, un système d'exploitation utilise un (voire plusieurs) système(s) de fichiers (file system). C'est le système de fichiers qui détermine les structures internes utilisées pour organiser les fichiers.

Parmi les nombreux systèmes de fichiers, citons :

- FAT, VFat, HPFS, NTFS (Dos & Windows),
- Ext, Ext2, Ext3, Ext4, ReiserFS (Linux)
- ISO9660, UDF, UFS, Joliet, RockRidge (pour les CD),
- SystemV, VxFS, Spiralog (Solaris, VMS)
- NFS (Network File System : pour les réseaux)

©EPFL 2024
J. Sam

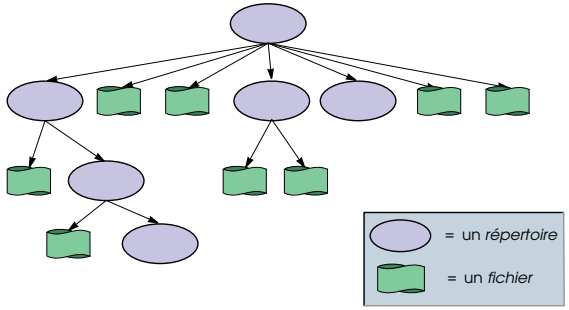
Structuration d'un système de fichiers

Grand nombre de fichiers

☞ fournir un moyen pour organiser ces fichiers

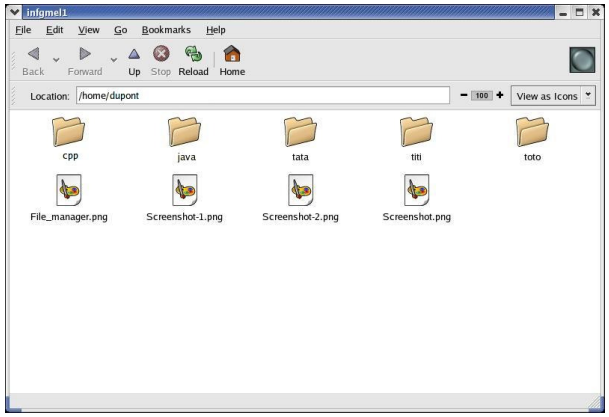
☞ concept de **répertoire** (directory)

Un répertoire est une collection (généralement non ordonnée) de fichiers ou de répertoires (alors appelés sous-répertoires).
Ils permettent d'organiser l'ensemble des fichiers dans une **structure arborescente**



©EPFL 2024
J. Sam

Structuration d'un système de fichiers (2)



En plus de la notion de répertoire, la plupart des systèmes permettent également de définir des **liens symboliques** vers des fichiers ou des répertoires (soft links avec UNIX, ou raccourcis dans d'autres systèmes), qui permettent de définir des alias (i.e., autres noms)

☞ permet d'assouplir la structure d'arbre

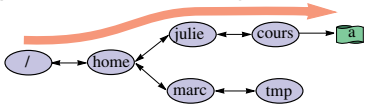
©EPFL 2024
J. Sam

Nommage des fichiers : absolu et relatif

On appelle **chemin** la succession des répertoires conduisant à un fichier, à partir d'un endroit donné dans l'arborescence.

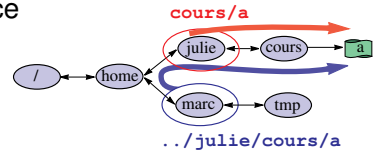
Pour désigner un fichier, il est possible de procéder de deux manières :

- à l'aide d'un **chemin absolu** : on prend comme convention un parcours de l'arbre partant de la racine
  Dans le cas de plusieurs arbres (forêt), le nom du lecteur est tout d'abord spécifié (i.e. on désigne la racine de l'arbre).



`/home/julie/cours/a`

- à l'aide d'un **chemin relatif** : c'est la succession des répertoires à traverser, à partir d'un autre répertoire de l'arborescence



`../julie/cours/a`

©EPFL 2024
J. Sam

Nommage des fichiers (2)

Le répertoire parent d'un sous-répertoire est désigné par `..`, tandis que le répertoire lui-même est désigné par `.`

Là y'a un point et là deux

Exemples de noms de fichiers (chemins) :

```
/home/prof/Work/cours/Info1/introduction2.tex
../images/paysages.gif
../../../toutlahaut.ps.gz
```

Sous UNIX/Linux, le délimiteur entre nom de répertoire et nom de fichier dans les chemins est la barre oblique slash : `/`

D'autres systèmes utilisent l'antislash ou backslash : `\`

```
D:\Users\Himher\Personnal Documents\introduction2.pdf
```

©EPFL 2024
J. Sam

Système de fichiers UNIX/Linux

Chaque utilisateur possède un **répertoire personnel** (home directory) dans lequel il peut placer ses fichiers personnels. C'est la racine du sous-arbre réservé spécifiquement à un utilisateur

Les noms de fichiers possèdent généralement une extension, délimitée par un `.`

Là aussi il y a un point

Cette extension peut être utilisée pour indiquer la nature du fichier, c'est-à-dire l'application à laquelle il est associé. Contrairement à d'autres systèmes d'exploitation, sous UNIX/Linux les fichiers peuvent avoir 0, 1 ou plusieurs extension(s).

Exemples :

`serie1.cc` : fichier de code source C++

`cours-1.ps.gz`

1re extension indiquant un fichier Postscript

2e extension indiquant un fichier compressé avec `gzip`

©EPFL 2024
J. Sam

Fichiers « cachés »

On distingue les fichiers/répertoires « cachés » au moyen d'une **convention** de nommage : ils sont préfixés par un `.`

Encore un point !

Exemple : `.cshrc`

Ce sont des fichiers/répertoires dont l'utilisateur n'a pas besoin explicitement (ou pas souvent), souvent des fichiers de configuration.

Pour voir les fichiers/répertoires « cachés », utilisez la commande

```
ls -a
```

©EPFL 2024
J. Sam

Fichiers et shell

Un certain nombre des fonctions du shell sont relatives au système de fichiers :

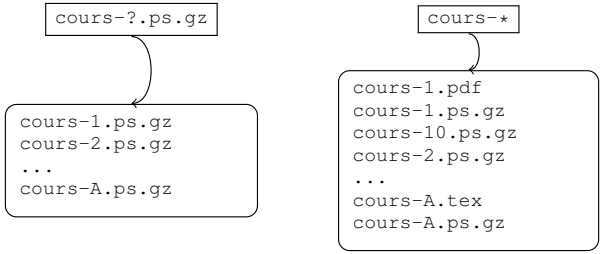
- Navigation dans la structure des fichiers :
  - répertoire courant (`pwd`)
  - modification de ce répertoire (`cd` = change directory),
  - lister le contenu d'un répertoire (`ls`),
  - copier des fichiers (`cp`) et les déplacer (`mv`),
  - effacer des fichiers (`rm`),
  - créer des liens (`ln`), etc...

Toutes les commandes soumises au shell sont interprétées relativement au **répertoire courant**.

©EPFL 2024
J. Sam

Fichiers et shell (2)

Les caractères de substitution (ou expressions régulières) permettent de spécifier plusieurs fichiers en une seule formule
`?` : remplace un seul caractère arbitraire
`*` : remplace une séquence quelconque de caractères
`[ ]` : remplace un seul caractère parmi ceux entre crochets.

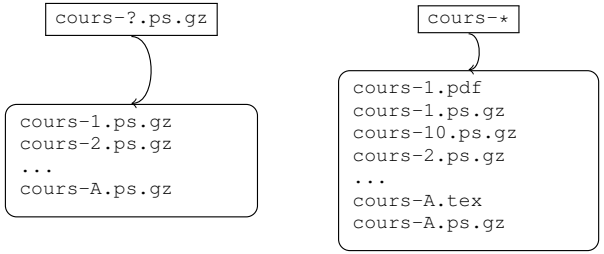


©EPFL 2024
J. Sam

Système d'exploitation
Shell
Système de fichiers
Editeurs et EDI

Fichiers et shell (2)

Les caractères de substitution (ou expressions régulières) permettent de spécifier plusieurs fichiers en une seule formule
`?` : remplace un seul caractère arbitraire
`*` : remplace une séquence quelconque de caractères
`[ ]` : remplace un seul caractère parmi ceux entre crochets.



©EPFL 2024
J. Sam

Système d'exploitation
Shell
Système de fichiers
Editeurs et EDI

Caractères de substitution Exemples

`ls ???`	liste tous les fichiers de trois lettres du répertoire courant
`ls *.txt`	liste tous les fichiers du répertoire courant se terminant par `.txt`
`ls *.[ch]`	liste tous les fichiers du répertoire courant se terminant par `.c` ou par `.h`
`ls *.cc *.h`	liste tous les fichiers du répertoire courant se terminant par `.cc` ou par `.h`

©EPFL 2024
J. Sam

Système d'exploitation
Shell
Système de fichiers
Editeurs et EDI

Méta-caractères

`*`, `?`, `[`, `]` sont donc des **caractères réservés** (ou « méta-caractères »), en ce sens qu'ils ont un rôle particulier dans l'interpréteur de commandes.

Il en existe d'autres : `| & ; ( ) < > $ \ " ' ~ ` `

Si on souhaite les utiliser, il faut les **protéger** ce qui se fait avec le « backslash » : `\`

Exemple :
`echo \\ abc \; \<`
affiche `\ abc ; <`

©EPFL 2024
J. Sam

Attributs de fichiers

Les attributs typiques d'un fichier sont :

- son nom
- sa taille
- la date et heure de création
- le propriétaire (créateur)
- les droits d'accès des autres utilisateurs
- modifiable, exécutable, caché,...
- système, possédant des alias,...
- la date et l'auteur de la dernière révision,
- ...

Attributs de fichiers (2)

Dans le cas du système Unix :

- A chaque fichier est associé un utilisateur propriétaire (le créateur du fichier) et un groupe propriétaire (l'un des groupes auxquels appartient l'utilisateur) : essayez la commande `ls -lF`.
- Les droits d'accès définissent trois attributs, et sont paramétrables pour 3 classes d'utilisateurs :

Attributs :	Classes d'utilisateurs :
Visibilité (lecture)	Propriétaire (owner ou user)
Modification (écriture, effacement)	Groupe (group)
Exécution	Autres (others)

Attributs de fichiers (3)

`ls -lF` (`l` : c'est un L minuscule, pas le chiffre un.)

Type de fichier : `d` = répertoire, `l` (L minuscule) = soft-link

```
drwx--x--x  2 sam devlp    4096 Sep 24 13:31 clipart/
-rw-------  1 sam users     675 Sep 10 18:17 cours01.ps.gz
drwx------  2 sam users    4096 Sep 24 11:50 CVS/
drwx--x--x  2 sam devlp    4096 Sep 24 13:27 figs/
-rw-------  1 sam users    1082 Sep 25 22:47 generaldef.sty
lrwxrwxrwx  1 sam users      38 Sep 10 18:17 make4print.sh -> ../m4p.sh*
-rw-------  1 sam users    5095 Sep 24 13:26 Makefile
-rw-------  1 sam users    1563 Sep 24 11:49 operateurs.tex
-rw-------  1 sam users   25093 Sep 24 13:34 prog3-C01.tex
-rw-r--r--  1 sam dilia 745110 Sep 25 21:00 prog3-intro.ps
-rw-------  1 sam users  25668 Sep 25 22:59 prog3-intro.tex
-rw-------  1 sam users    1028 Sep 24 11:44 variables.tex
```

droits — user — groupe — taille (par défaut en octets) — date (dernière modification) — nom

Attributs de fichiers (4)

Syntaxe :

`chmod` `(u|g|o|a)*` `+|-` `(r|w|x)*`

- `(u|g|o|a)*` : User, Group, Other, All (`a`=`ugo`)
- `+|-` : `+` : ajouter, `-` : supprimer
- `(r|w|x)*` : Read, Write, eXecute

(*) c'est-à-dire au moins un!

chmod o-r
chmod go+rx
chmod a+w

Exemples :

```
chmod a+x monscript.sh
chmod go-r perso
```

☞ tout le monde peut exécuter ce fichier
☞ les autres que moi n'ont pas le droit de lire ce fichier/repertoire

Editeurs de texte

Pour écrire et modifier des fichiers le moyen le plus naturel est d'utiliser un un *éditeur de texte* tels que :

- Atom
- Emacs
- Geany
- Gedit
- Sublim Text
- Vim
- ...

Connaître un/des éditeur(s) de texte est absolument indispensable !

☞ La mini-référence «Environnement Unix», disponible sur le site du cours dédie une de ses sections à Emacs et Geany. Vous aurez aussi l'occasion de les utiliser en TP.

©EPFL 2024
J. Sam

IntelliJ et Eclipse

IntelliJ et Eclipse sont des environnements de développement intégrés (EDI)
Il en existe beaucoup d'autres :

- netbeans, kdevelop, ...

IntelliJ est conseillé ce semestre

☞ Attention non disponible sur les machines de l'EPFL

Un environnement de développement intégré est un outil très puissant, qui facilite la **rédaction**, la **mise à jour** et la **correction** de programmes, mais qui est parfois aussi très «lourd» pour de petits projets.

Pour créer ou modifier de simples programmes, il est parfois beaucoup plus rapide de passer par un simple éditeur de texte !

©EPFL 2024
J. Sam