Unix et Programmation Web

Cours 1

kn@lri.fr

Contenu du cours

1. Comprendre les bases du Web
   - Fonctionnement des ordinateurs Unix (cours 1 et 2)
   - Notions de réseau (cours 3 et 4)
2. Programmer (pour) le Web
   - Pages statiques (HTML & CSS cours 5)
   - Pages dynamiques avec PHP (cours 6, 7 et 8)
   - Notions de sécurité des sites Web (cours 9)
   - Bonus (cours 10)

Cours disponible en ligne sur https://www.lri.fr/~kn/teaching_fr.html. Les supports de cours seront distribués à partir de la semaine prochaine.

Modalités de Contrôle des Connaissances (MCC)

2 sessions:

- 1ère session
  - Contrôle continu (50%):
    - Partiel (fin octobre/début novembre) 25%
    - 1 interro(s) de TP (25 %)
  - Examen (50%)
- 2ème session (examen 100%)

Rappel: *La défaillance fait obstacle au calcul de la moyenne et implique l'ajournement. La présence de l'étudiant étant obligatoire en TP [...], plus d'une absence injustiée dans un enseignement peut entraîner la défaillance de l'étudiant dans l'enseignement concerné*

Organisation

- 10 semaines de cours :
  - 8/9, 16/9, 23/9, 30/9, 7/9, 14/9
  - partiel/toussaint
  - 4/11, (pas de cours le 11/11), 18/11, 25/11, 2/12
- 10 semaines de TP :
  - 18/9, 25/9, 2/10, 9/10, 16/10
  - partiel/toussaint
  - 6/11, 13/11, 20/11, 27/11, 4/12
- examen début Janvier
- cours tous les mardi (13h30)
- TP le jeudi (9h)

Plan

Système d'exploitation

Quelques systèmes:

- Windows XP/NT/2003/7/8, ...
- Linux, FreeBSD, NetBSD, OpenBSD, ...
- MacOS X (basé sur une variante de FreeBSD), ...
- Unix, AIX, Solaris, HP-UX, ...
- Symbian OS (Nokia), iOS, Android, ...

Système d'exploitation

Qu'est-ce qu'un système d'exploitation ?

- c'est un programme
- qui organise l'accès aux ressources de la machine

Quelles sont les ressources d'une machine?

- Processeur (temps d'exécution)
- Mémoire
- Accès aux périphériques de stockage
- Accès aux périphériques d'entrées/sorties
- ...

Système d'exploitation

Haut niveau

Bas niveau

Applications: navigateur Web, éditeur de texte, anti-virus, jeu, compilateur, ...

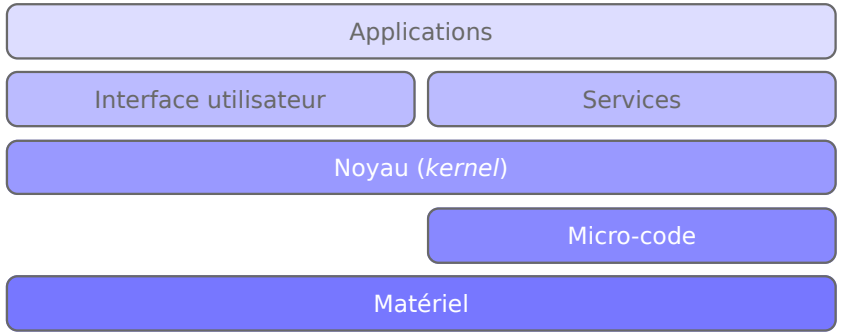
Système d'exploitation:
- Gestion des ressources
- Interface avec le matériel (pilotes)

Matériel: CPU, mémoire, périphériques, ...

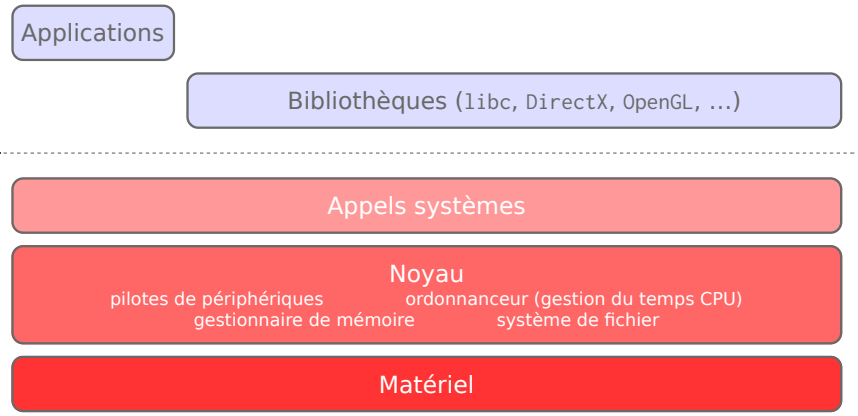
Le système Unix

1965 : MultICS: *Multiplexed Information and Computing Service* (Bell & MIT)
1969 : Unix: 1ère version en assembleur (AT&T)
1972-73 : Unix réécrit en C
1976 : Invention de TCP/IP
1977 : *Berkeley Software Distribution* (BSD)
1985 : Unix System V
1988 : Minix
1992 : Linux

Unix : architecture



Zoom sur le noyau



Le *Shell* Unix

- Interface utilisateur en mode texte
L'utilisateur écrit des commandes dont le résultat est affiché à l'écran
- Interface « historique » sous Unix
- Expose à l'utilisateur certains concepts Unix (permissions, propriétaires, processus, ...)
- Ces concepts sont importants pour pouvoir concevoir de sites Web

Le *Shell* Unix

Exemple de session *shell*:

```
$ ls
Documents   Downloads   Public    Person
$ cd Documents
$ ls
compte_rendu.txt
$ mv compte_rendu.txt cr.txt
$ ls
cr.txt
```


Plan

Système de gestion de fichiers (*filesystem*)

- Organise les données sur le support physique
- Protège contre les corruptions de données
- Gestion optimale de l'espace disponible
- Accès efficace aux données
- Abstraction du support physique (DVD, mémoire flash, disque réseau, ...)
- Enregistrement des méta-données (date de création, propriétaire, taille, ...)

Le concept de fichier

Un fichier est une collection d'infromations numériques réunies sous un même nom et enregistrée sur un support de stockage

- Manipulable comme une unité
- Selon les systèmes, le nom a plus ou moins d'importance
- possède un type

Le concept de fichier

Ne pas confondre:

- type du fichier: il influe sur le comportement du système (fichier « normal », répertoire, lien (raccourcis), fichier système, ...). C'est une méta-donnée conservée par le système de fichier
- type du contenu: le type des données contenues dans le fichier:
  - DOS puis Windows: l'extension (les 3 derniers caractères après le « . ») détermine le type de contenu
  - MacOS puis OS X/iOS: les premiers octets du fichier déterminent son type
  - Premiers octets ou extension, selon les interfaces utilisées

Les attributs d'un fichier

Nom :
Propriétaire : utilisateur qui possède ce fichier
Groupe : groupe d'utilisateurs qui possède ce fichier
Emplacement : localisation du fichier sur le support physique
Taille : en octet (peut être la taille réelle ou la taille occupée sur le support)
Permissions : « qui a quel droit » sur le fichier (lecture, écriture, exécution, ...)
Type :
Dates : dernier accès, dernière modification, création, ...

Organisation logique des fichiers

Usuellement, les fichiers sont regroupés en répertoires. Les répertoires sont imbriqués les uns dans les autres de manière a former une arborescence.

Sous Unix il y a un répertoire racine, « / » (*slash*) qui contient toute l'arborescence du système.
Chaque utilisateur possède aussi un répertoire personnel

Noms de fichiers et chemins

Un chemin est une liste de répertoire à traverser pour atteindre un fichier ou répertoire donné. Sous Unix, le séparateur de chemin est le « / »
Les chemins absolus commencent par un `/` et dénotent des fichiers à partir de la racine. Exemple:

```
/home/kim/Documents/ProgInternet/cours01.pdf
```

Les chemins relatifs dénotent des fichiers à partir du répertoire courant. Exemple:

```
Documents/ProgInternet/cours01.pdf
```

si on se trouve dans le répertoire `/home/kim`

Les noms spéciaux: « . » dénote le répertoire courant, « .. » le répertoire parent, « ~ » le répertoire de l'utilisateur et « `~toto` » le répertoire de l'utilisateur `toto`

Utilisation du *Shell*

Le shell affiche un invite de commande (*prompt*). Exemple:
`kim@machine $`

On peut alors saisir une commande:

```
kim@machine $ ls *.txt
```

Le shell affiche la sortie de la commande:

```
fichier1.txt   fichier2.txt
```

Certains caractères doivent être précédés d'un « \ » (échappés):

```
kim@machine $ ls mon\ fichier\#1.txt
```


La ligne de commande

Une ligne de commande a la forme:
`prog item1 item2 item3 item4 …`

1. Si `prog` est un chemin il doit dénoter un fichier exécutable
2. Si `prog` est un simple nom, il doit dénoter un fichier exécutable se trouvant dans un des répertoires prédéfinis (`/bin`, `/usr/bin`, ...)
3. Pour chaque $\texttt{item}_i$ (séparés par un ou plusieurs espaces non échappés) le *shell* fait une expansion de nom
4. La liste de toutes les chaînes de caractères expansées est passée comme argument au programme `prog`

Expansion des noms
Expressions régulières glob

Certains caractères sont interprétés de manière spéciale par le *shell*. Ces caractères sont « expansés » selon des règles. Si la forme expansée correspond a un ou plusieurs fichiers existants, alors leurs noms sont placés sur la ligne de commande. Sinon la chaîne de caractère de départ garde sa valeur textuelle.

Expressions régulières glob

Règles d'expansion: `*` n'importe quelle chaîne
`?` n'importe quel caractère `[ab12…]` un caractère dans la liste
`[^ab12…]` un caractère absent de liste
`[a-z]` un caractère dans l'intervalle
`[^a-z]` un caractère absent de l'intervalle
$\{m_1, m_2\}$ motif m_1 ou m_2
$?(m_1|\ldots|m_n)$ $@(m_1|\ldots|m_n)$ $*(m_1|\ldots|m_n)$ $+(m_1|\ldots|m_n)$
k motifs parmi m_i
?: $0 \leq k \leq 1$ @: $k = 1$ *: $k \geq 0$ +: $k \geq 1$
$!(m_1|\ldots|m_n)$: ni m_1, ..., ni m_n

Expressions régulières glob
Exemples

`ls !(*[aeiouy]?)` La chaine « `!(*[aeiouy]?)` » est remplacée par la liste de tous les fichiers dont l'avant dernière lettre du nom n'est pas une voyelle. S'il n'y a pas de tel fichier, la chaîne « `!(*[aeiouy]?)` » est passée à la commande `ls`.

`ls [0-9]*` affiche la liste des fichiers commençant par un chiffre

`ls +(abc)` affiche la liste des fichiers dont le nom est une répétition de « abc ».

Commandes shell de base

cd *chemin*: *chemin* devient le répertoire courant. Si absent, utilise le répertoire personnel

ls $chemin_1$ … $chemin_n$: affiche le nom des *n* fichiers. Si *n=0* affiche le contenu du répertoire courant. Avec l'option `-l` affiche la liste détaillée.

cp $chemin_1$ $chemin_2$: copie de fichier

mv $chemin_1$ $chemin_2$: déplacement de fichier (et renommage)

rm $chemin_1$ … $chemin_n$: supprime les fichiers (définitif)

Droits et propriétés des fichiers

Sous Unix un utilisateur est identifié par son login (ou nom d'utilisateur).
Chaque utilisateur est dans un groupe principal.
Chaque fichier appartient à un utilisateur et à un groupe.
Chaque fichier possède 3 permissions pour son propriétaire, son groupe et tous les autres. Les permissions sont lecture, écriture, exécution (plus d'autres non abordées dans ce cours).

Permission	**fichier**	**répertoire**
lecture (r)	lire le contenu du fichier	lister le contenu du répertoire
écriture (w)	écrire dans le fichier	supprimer/renommer/créer des fichiers dans le répertoire
exécution (x)	exécuter le fichier (si c'est un programme)	rentrer dans le répertoire

```
$ ls -l
drwxr-x---  9 kim prof 4096 Sep  7 21:31 Documents
```


La commande chmod

chmod *permissions* $chemin_1$ … $chemin_n$

modifie les permissions des fichiers *1* à *n*. La chaîne *permissions* est soit une suite de modifications de permissions symbolique soit l'ensemble des permissions données de manière numérique:

```
chmod 755 fichier.txt
chmod u-w,a+x,g=w fichier.txt
```

Permissions numériques

On groupe les bits de permissions par trois puis on convertit en décimal:

Utilisateur			Groupe			Autres		
r	w	x	r	w	x	r	w	x
1	1	0	1	0	0	0	0	0
6			4			0		

Le fichier est lisible et modifiable mais pas exécutable par son propriétaire, lisible pour le groupe. Les autres ne peuvent ni le lire ni le modifier.

Permissions symboliques

```
cible modifieur permission
```

cible : u (utilisateur), g (groupe), o (others), a (all)
modifieur : + (autorise), - (interdit), = (laisse inchangé)
permission : r (lecture), w (écriture), x (exécution)

Exemple:

```
chmod u+rw,u-x,g+r,g-wx,o-rwx fichier.txt
```


Liens symboliques (1)

Pour des raisons d'organisation, on veut pouvoir « voir » le même fichier ou répertoire sous deux noms différents (ou à deux endroits différents). Par exemple:

```
$  ls -l Documents/Cours
total 8
drwxr-xr-x 3 kim prof 4096 Sep  9 11:30 Licence
drwxr-xr-x 3 kim prof 4096 Sep  9 11:30 Master

$  cd Documents/Cours/Master; ls
 Compilation    XMLProgInternet

$  cd XML_Prog_Internet; ls
cours01 cours02  cours03 cours04 cours05  cours06   Prereq

$ ls -l Prereq
lrwxrwxrwx 1 kim prof 28 Sep  9 11:30 Prereq -> ../../Licence/UnixProgWeb/
```


Liens symboliques (2)

La commande `ln` permet de créer des liens symboliques. Un lien est un petit fichier qui contient un chemin vers un fichier de destination.

Exemple d'utilisation

```
$ ln -s ../foo/bar/baz/toto.txt rep/titi.txt
```

crée un lien vers le fichier `toto.txt` sous le nom `titi.txt` (chacun placé dans des sous/sur répertoires)

- Ouvrir/modifier le lien > ouvre/modifie la cible
- Supprimer le lien > supprime le lien mais pas la cible
- Si la cible est un répertoire, faire `cd` nous place « dans » la cible, mais le repertoire parent est celui d'où l'on vient

Cela permet de créer l'illusion que la cible a été copiée à l'identique, sans les inconvénients

À propos de la suppression

La commande `rm fichier` efface un fichier définitivement
La commande `rm -d rep` efface un répertoire s'il est vide
La commande `rm -r rep` efface un répertoire récursivement mais demande confirmation avant d'effacer des éléments
La commande `rm -rf rep` efface un répertoire récursivement et sans confirmation

Toute suppression est définitive

Gag classique :

```
$  mkdir \~
...
$ ls
  Documents  Photos  Musique   ~
$ rm -rf ~
```

Obtenir de l'aide sur une commande

La commande `man` permet d'obtenir de l'aide sur une commande. Lors qu'une page d'aide est affichée, on peut la faire défiler avec les touches du clavier, la quitter avec « `q` » et rechercher un mot avec la touch « `/` »

```
LS(1L)            Manuel de l'utilisateur Linux            LS(1L)


NOM
       ls, dir, vdir - Afficher le contenu d'un répertoire.

SYNOPSIS
       ls [options] [fichier...]

       Options POSIX : [-1acdilqrtuCFR]

       Options  GNU  (forme courte) : [-1abcdfgiklmnopqrstuxABCD
       FGLNQRSUX] [-w cols] [-T cols]  [-I  motif]  [--full-time]
       [--format={long,verbose,commas,across,vertical,single-col
       umn}]                  [--sort={none,time,size,extension}]
       [--time={atime,access,use,ctime,status}]
       [--color[={none,auto,always}]] [--help] [--version] [--]

DESCRIPTION
       La commande ls affiche  tout  d'abord  l'ensemble  de  ses
       arguments  fichiers  autres  que  des répertoires. Puis ls
       affiche  l'ensemble  des  fichiers  contenus  dans  chaque
       répertoire  indiqué.   dir et vdir sont des versions de ls
       affichant par défaut leurs résultats  avec  d'autres  for
       mats.
```


Recherche de fichiers

La commande `find rep` *`criteres`* permet de trouver tous les fichiers se trouvant dans le répertoire `rep` (ou un sous répertoire) et répondant à certains critères. Exemples de critères :

- `-name '*toto*'` dont le nom contient `toto`
- `-size +200M` dont la taille sur le disque est supérieure à 200 Mo
- c_1 `-a` c_2 pour lesquel les critères c_1 et c_2 sont vrais
- c_1 `-o` c_2 pour lesquel l'un au moins des critères c_1 et c_2 est vrais
- `-user toto` qui appartiennent à l'utilisateur `toto`

Comment trouver toutes les options de la commande `find` ? `man find`

Quelques commandes utiles

- `cat` *`fichier`* permet d'afficher le contenu d'un fichier dans le terminal
- `less` *`fichier`* permet de lire le contenu d'un fichier (avec défilement en utilisant les flêches du clavier si le fichier est trop grand)
- `sort` *`fichier`* permet d'afficher les lignes d'un fichier triées (on peut spécifier des options de tri)
- `file` *`fichier`* permet de connaître le type d'un fichier
- `wc` *`fichier`* permet de compter le nombre de caractères/mots/lignes d'un fichier
- `head` *`fichier`* permet de garder les n premières lignes d'un fichier

On verra comment composer ces commandes pour exécuter des opérations complexes