Partie I : Introduction à Linux

1 Découverte de l'environnement de travail

Durant vos études à *Polytech'Nice-Sophia*, vous allez travailler sous deux systèmes d'exploitation différents :

- Unix (Linux sur PC)
- Windows XP (sur PC)

Ce TD est une introduction au système Unix qui va être cette année votre principal environnement de travail.

1.1 Se connecter

Sous Unix, chaque utilisateur est identifié par le système grâce à un numéro individuel: l'*UID*. Pour des raisons de convivialité, une chaîne de caractères est associée à ce numéro (par exemple votre nom). Lors de la phase de connexion, le système identifie l'utilisateur grâce à une combinaison *nom d'utilisateur / mot de passe*. C'est cette association qui permet d'assurer la confidentialité de vos données ainsi que de vous identifier de manière unique. Il est donc impératif que votre mot de passe soit confidentiel et non trivial. Un *login* et un mot de passe vous ont été fournis pour vous connecter sous les systèmes du département SI de *Polytech'Nice-Sophia*.

Sous Unix, plusieurs environnement graphiques coexistent. Par défaut, vous serez sous l'environnement KDE sous Linux. Si cet environnement ne vous plaît pas, vous pouvez en changer au moment de la saisie de votre login/password.

1.2 Se déconnecter

- Pour vous déconnecter sous KDE, choisissez *déconnexion* dans le menu **K** en bas à gauche de la barre des tâches.
- Pour vous déconnecter d'une *session* Unix (dans un terminal), utilisez la commande **exit**.
- Dans un environnement graphique, vous pouvez avoir recours à `xlock` pour interdire temporairement l'accès à votre poste de travail sans vous déconnecter. Notez toutefois que l'utilisation de cette commande doit être limitée dans le temps, d'autres étudiants pouvant avoir besoin de la machine pendant votre absence. En particulier, l'utilisation immodérée de cette commande pourra vous valoir un déconnexion brutale (avec les risques que cela entraîne) de la part des enseignants.

1.3 Les serveurs

Le département SI dispose de deux serveurs Web : un serveur public qui est accessible depuis l'extérieur et un serveur local qui n'est accessible que des machines du département. Leurs adresses respectives sont `http://www.essi.fr` et `http://www-local.essi.fr`.

Allez lire les informations générales sur le département : Docs & textes officiels, règlement intérieur. Allez lire aussi les FAQ (Foire Aux Questions) avant de poser une question, pour être sur que la réponse n'existe pas déjà !

Un autre service particulièrement utile pour votre vie de tous les jours au département SI est l'emploi du temps. Il est accessible à l'URL `http://edt`.

1.4 Le courrier électronique

Vous disposez de plusieurs clients de messagerie mais nous vous conseillons d'utiliser Kmail le client de KDE. Pour configurer votre client de messagerie vous avez besoin des informations suivantes :

- pour le courrier entrant :
  - type de connection : IMAP
  - serveur de mail entrant : **mail.essi.fr**
  - méthode d'encryption : SSL

- le courrier sortant :
  - protocole de transport : SMTP
  - serveur de mail sortant : **smtp.essi.fr**

Pour configurer Kmail :

- lancez Kmail (menu K, sous-menu Internet)
- choisissez *Configure Kmail* dans le menu *Settings*
- saisissez les informations précédentes dans les rubriques *Identities* et *Accounts*

Une fois configuré, testez votre client de messagerie en composant un nouveau message que vous adresserez à un de vos collègues. Mettez vous en copie du message et vérifiez que vous recevez bien cette copie.

Remarques:

- votre adresse email à *Polytech'Nice-Sophia* est : **login@essi.fr** (où *login* est le nom que vous donnez au login). Quand vous êtes sur une machine du département SI (domaine **essi**) et que vous envoyez un mail à une personne du même domaine, il suffit de donner le nom d'utilisateur.
- Prenez l'habitude de lire vos mails très régulièrement car c'est la façon qu'utiliseront les enseignants ou l'administration pour vous contacter de manière individuelle.

2 Documentation

La syntaxe générale des commandes Unix est la suivante :

cmd [-opt1] [-opt2] [...] [-optN] [param1] [param2] [...] [paramN]

Plusieurs programmes vous permettent d'obtenir de la documentation sur un sujet :

- **apropos** : permet d'obtenir toutes les commandes ayant un rapport avec un sujet. Faites quelques tests pour cette commande (ex `apropos` password, `apropos` login, ...)
- **man** : fournit la page de manuel de la commande donnée en paramètre. Testez cette commande (ex `man` date, `man` login, ...). Les pages de manuels sont organisées en sections (Voir cours). Si une même commande existe dans plusieurs sections, vous pouvez spécifier la section que vous voulez : `man` *n* `cmd`
- la documentation relative à ce cours : visitez la page d'accueil de ce cours à l'URL

  `http://www-local.essi.fr/~gaetano/unix`

2.1 Exercices

- Visualisez les pages de manuel de quelques commandes vues en cours.
- Recherchez la commande affichant le calendrier du mois en cours.
- Recherchez le jour (dans la semaine) de votre naissance

3 Le système de fichiers Unix

3.1 Qu'est-ce qu'un fichier

Un fichier est une chaîne de caractères non structurée : le système ne possède aucune notion d'organisation de fichiers (séquentielle, indexée, ...). A tout fichier est associé un bloc, nommé i-nœud (index-node ou i-node) contenant un certain nombre d'informations d'ordre général concernant le fichier :

- sa taille
- l'adresse des blocs utilisés sur le disque pour ce fichier
- l'identification de son propriétaire et ses droits d'accès par les différentes classes d'utilisateurs
- son type (fichier ordinaire, répertoire, ou fichier spécial)
- trois dates relatives aux opérations effectuées sur ce fichier (création, dernière consultation, dernière modification)

3.2 Chemin relatif ou absolu

Le système de fichiers est hiérarchisé sous la forme d'un arbre. La racine de cet arbre est /. Pour tout fichier, il existe un chemin de la racine vers ce fichier. La liste des noms de répertoires situés depuis la racine jusqu'au fichier considéré définit le chemin d'accès absolu à ce fichier. Mais les fichiers sont aussi accessibles même si l'on ne connaît pas le chemin absolu ; on peut y accéder en se déplaçant par rapport à l'endroit où l'on se trouve : c'est le chemin relatif (qui ne commence donc pas par /).
Comment trouver son chemin dans cette hiérarchie de fichiers ? Les commandes **pwd**, **cd**, **locate** doivent vous permettre de répondre aux questions suivantes :

- Lorsque vous vous connectez, vous êtes sous votre répertoire initial de travail. Quel est le nom absolu de ce répertoire ?
- Quel est le nom relatif de celui de votre voisin ?
- Déplacez vous dans ce répertoire.
- Pensez au raccourci pour un retour rapide "à la maison" (cd ~).
- Trouvez où est rangé le programme `emacs`.

3.3 Exercices

Les principales commandes utilisées dans cet exercice sont : **ls**, **mkdir**, **rmdir**, **cp**, **rm**, **mv**, **chmod** et **ln**.

- Dans votre répertoire; affichez la liste de tous les fichiers :
  - sous un format "condensé"
  - avec un format long (donnant le propriétaire, les permissions, la taille, ...)
  - y compris les fichiers cachés (commençant par un point)
  - avec un format long, mais dans l'ordre alphabétique inverse
  - avec un format long, mais du plus récent au plus ancien
  - avec un format long, mais du plus ancien au plus récent
- Créez un répertoire *system* sous votre répertoire initial de travail, puis un répertoire *tp1* et *tp2* sous *system*
- Renommez le répertoire *system* en *linux*
- Effacez le répertoire *tp2*
- Essayez de déplacer un fichier de /bin dans *tp1*. Que se passe-t-il ?
- Copiez un fichier de /bin dans *tp1*
- Effacez tout ce se qui se trouvre sous *linux* en une seule commande
- Dans le répertoire *linux*, créez les fichiers de noms exacts *fich1*, *fich2*, *fich11*, *fich12*, *fich1a*, *ficha1*, *fich33*, *.fich1*, *.fich2*, *afich* et *toto*, puis listez les fichiers :
  - dont les noms commencent par *fich*
  - dont les noms commencent par *fich* suivi d'un seul caractère
  - dont les noms commencent par *fich* suivi d'un seul caractère qui est un chiffre
  - dont les noms commencent par .
  - dont les noms ne commencent pas par *f*
- Créez le fichier vide *fichier* (avec `touch`) dans *tp1*. Examinez les permissions de *fichier*
- Changez les permissions du répertoire *linux* pour :
  - supprimer tous les accès sauf pour vous
  - ajouter l'accès en lecture, écriture et exécution pour le propriétaire seul
  - ajouter l'accès en lecture et en exécution pour le groupe

- Créez dans le nouveau répertoire *linux/tmp* le fichier *perm*, puis positionner les permissions de *perm* pour que vous puissiez :
  - le lire, le modifier et le supprimer
  - le lire et le modifier mais pas le supprimer
  - le lire et le supprimer mais pas le modifier
  - le lire mais ni le modifier et ni le supprimer
- Créez le fichier *share* dans le répertoire *linux/tmp* et modifiez les droits d'accès nécessaires pour que votre voisin (qui est dans votre groupe) :
  - puisse lire et modifier le fichier *share*, sans toutefois pouvoir le supprimer
  - ne puisse rien faire d'autre sur les répertoires *linux* et *linux/tmp*
- Utilisez successivement les commandes **cat**, **more**, et **less** sur des fichiers cachés de votre répertoire initial de travail.
- Créez sous **xemacs** le fichier `~/linux/tp1/liste` contenant la liste des fichiers de votre répertoire racine
- Créez un lien symbolique de `~/linux/tp1/liste` vers `~/linux/tp1/liste.sym`
- Créez un lien physique de `~/linux/tp1/liste` vers `~/linux/tp1/liste.phys`
- Vérifiez que ces deux nouveaux fichiers ont bien le même contenu que le fichier `~/linux/tp1/liste`
- Renommez le fichier `~/linux/tp1/liste` en `~/linux/tp1/new`
- Le contenu des fichiers `liste.phys` et `liste.sym` est il toujours le même ? Pourquoi ?

4 Outils divers

4.1 Conversion de fichiers pour l'impression

Les formats de fichiers imprimables les plus courants concernant sont :

- ASCII (*plain text*), visualisable avec tout éditeur de texte Linux (`xemacs`, `ed`, `vi`, ...)
- PostScript (`.ps`), visualisable avec les commandes **ghostscript**, **ghostview**, **gv**, qui simulent en fait une imprimante postscript.
- Portable Document Format (`.pdf`), visualisable par **acroread** ou **gv**.

Les utilitaire permettant de passer d'un format à l'autre sont : **a2ps**, **ps2ascii ps2pdf**, **pdf2ps**.

- Créez un fichier PostScript à partir d'un fichier texte ASCII quelconque (voir l'option `-o` de `a2ps` sinon, le fichier généré est directement envoyé à l'imprimante).
- Créez un fichier PostScript à partir d'un fichier source C avec la commande **a2ps**.
- Visualisez le fichier `.ps` généré.
- Recommencez en produisant cette fois un fichier PostScript à raison d'une page (de source ASCII) par page (PostScript) dans lequel les lignes de code sont numérotées.
- Cherchez un document au format PDF sur le Web (pas trop gros). Sauvez-le dans votre répertoire de travail et produisez une version ASCII de ce fichier.

4.2 Commandes d'impression

Trois commandes vous permettent de gérer l'impression de fichiers PostScript :

- `lpr -P`*Printer* `fichier.ps` : imprime `fichier.ps` sur l'imprimante *Printer*.
- `lpq -P`*Printer* : liste les travaux en attente d'impression sur l'imprimante spécifiée.
- `lprm -P`*Printer Jobnumber* : permet de supprimer de la file d'attente le travail correspondant à *Jobnumber*. Vous ne pouvez bien sûr supprimer que des travaux qui vous appartiennent.

La valeur de la variable d'environnement `PRINTER` désigne l'imprimante par défaut, ce qui permet d'éviter la spécification du paramètre `-P` pour toutes ces commandes.
Remarque : Vous disposez d'un quota limité pour vos impressions. Ne faites donc pas des impressions inutiles.

4.3 Archivage et compression de fichiers

Grâce à la commande **tar**, vous pouvez créer des archives. Vous pouvez y inclure plusieurs fichiers. L'extension donnée à ces fichiers est `.tar`. Les options les plus utiles sont les suivantes :

- `c` : Création d'une nouvelle archive.
- `x` : Extrait les fichiers d'une archive.
- `t` : Liste le contenu d'une archive.
- `v` : Liste les fichiers sur lesquels le programme travail.
- `f <filename>` : Spécifie un nom d'archive.
- `z` : Travaille sur une archive compresée (sous Linux).

L'utilitaire **gzip** permet quant à lui de compresser un ou des fichiers (souvent des archives) :

- `-d` : Décompression au lieu de compression.
- `-9` : Niveau de compression maximum.

Exercices

- Faites une archive de votre répertoire de travail
- consultez le contenu de l'archive
- compressez cette archive
- consultez à nouveau le contenu de l'archive compressée
- essayez de décompresser cette archive dans votre répertoire `~/tmp`
  - en utilisant l'option `-z`
  - en utilisant explicitement la commande **gzip**
- échangez des archives compressées par mail avec votre voisin, en préservant les protections des fichiers originaux

4.4 L'éditeur XEmacs

Un *éditeur* est un programme qui permet de fabriquer un fichier, principalement un fichier de texte (souvent, ce fichier n'est qu'une étape intermédiaire avant la production du document final). Il existe plusieurs éditeurs disponibles sous Linux. Parmi les plus populaires, on peut citer **ed**, **vi** et **xemacs**. On peut trouver aussi des éditeurs similaires à Word de Microsoft, comme par exemple son concurrent direct OpenOffice. Pour éditer des fichiers ascii et en particulier des programmes sources (C, C++, Java, shell,...) l'éditeur **xemacs** reste la référence. Nous vous proposons ici d'aborder cet outil indispensable à l'aide d'une série de petits exercices. Allez à l'URL `http://www-local.essi.fr/~gaetano/unix` et cliquez sur `exercices sur XEmacs`.

5 Redirections et pipes

Tout processus Unix travaille avec un fichier standard d'entrée (*stdin*), un fichier standard de sortie (*stdout*) et un fichier standard d'erreur (*stderr*). Par défaut, ceux-ci correspondent au clavier et à l'écran ou terminal où est connecté l'utilisateur. Selon les cas, ces fichiers peuvent être redirigés soit vers un autre fichier (pour une sauvegarde par exemple), soit vers une autre commande.

Il est possible d'exécuter plusieurs commandes de façon séquentielle, sans que le résultat de l'une influe sur l'autre. Pour cela, on utilise le "**;**" pour séparer les commandes. L'exécution est purement séquentielle, c'est à dire que chaque commande ne commence son exécution que lorsque la précédente est terminée. Mais on peut souhaiter, au contraire, récupérer le résultat d'une commande pour qu'il serve d'entrée à une autre. On utilise alors le mécanisme de redirection. Les commandes fonctionnent alors en parallèle.

5.1 Quelques rappels

- Sur le flux d'entrée standard (*stdin* ou 0) :
  - `<` utilise le fichier passé comme entrée (ex : `cat < fichier`).
  - `<<` utilise l'entrée standard avec une marque de fin de fichier (ex : `cat <<END` va saisir son entrée jusqu'à ce que le mot `END` soit tapé seul en début de ligne).
- Sur le flux de sortie standard (*stdout* ou 1) :
  - `>` redirige dans un fichier en le créant.
  - `>>` concatène à la fin du fichier.
- Sur le flux d'erreur standard (*stderr* ou 2) :
  - `2>` et `2>>` avec les mêmes règles que pour `>` et `>>`.

Le caractère | (*pipe*) permet de transmettre le flux de sortie de la commande en partie gauche, en flux d'entrée de la commande en partie droite.

5.2 Exercices simples

Les exercices suivants utilisent les mécanismes de redirection et quelques commandes Unix fondamentales : **cat**, **wc**, **head**, **tail**, **grep** et **diff**. Placez-vous dans votre répertoire *linux/tp1*.

- Mettez la date dans le fichier *aujourdhui*
- Copiez le fichier */etc/termcap* dans le répertoire *tp1*
- Combien le fichier *termcap* comporte-t-il de lignes ? de mots ? de caractères ?
- Créez un fichier *debut* formé des 10 premières lignes de *termcap* et un fichier *fin* des 20 dernières
- Créez un fichier *extremes* constitué de la concaténation des fichiers *debut* et *fin*
- Renommez le fichier *extremes* en *extremes.old*
- Essayez de reconstruire le fichier *extremes*, mais en une seule ligne de commande sans utiliser les fichiers *debut* et *fin*, et vérifiez que les deux fichiers *extremes.old* et *extremes* sont identiques
- Lancez la commande `ypcat passwd | more`
- Créez sous tp1 le fichier *login.passwd* (où *login* correspond à votre nom utilisateur) qui contient la ligne du fichier *passwd* vous concernant
- Créez un fichier `liste1` contenant la liste des informations relatives aux utilisateurs du département ayant un nom ou un prénom contenant le mot `Jean`
- Créez un fichier `liste2` contenant la liste des informations relatives aux utilisateurs du département ayant **exactement** le prénom `Jean`

5.3 Exercices plus corsés

Pour certains des exercices qui suivent, considérez les nouvelles commandes **tr**, **cut**, **column** et **find**.

- Créez et effacez un fichier appelé `-i`
- Affichez le nombre de fichiers et sous-répertoires de votre répertoire initial
- Affichez, en les numérotant de 40 à 50, les lignes 40 à 50 du fichier */etc/termcap*
- Affichez la liste des sous-répertoires (et seulement des sous-répertoires) de votre répertoire initial
- Affichez la liste des fichiers de votre répertoire initial qui ne sont pas des sous-répertoires
- Affichez la taille des fichiers de votre répertoire initial qui ne sont pas des sous-répertoires sous forme lisible (par exemple `4.0K` au lieu de `4096`) et seulement la taille (pas d'autre information)
- Créez le fichier *users* donnant liste des utilisateurs du département triée par ordre alphabétique suivant les *login*s. Chaque ligne du fichier comprendra le *login*, l'*UID*, le nom et le prénom d'un utilisateur, séparés par un espace

- En utilisant la commande adéquate, trouvez et listez :
  - tous les fichiers ou répertoires sous le répertoire */etc* dont les noms commencent par *rc*. Vous ne devez afficher aucun message d'erreur
  - tous les fichiers réguliers vous appartenant. Mettez le résultat dans */tmp/mesfichiers*
  - tous les sous-répertoires de */etc*
- Affichez exclusivement tous les fichiers et répertoires cachés (fichiers et répertoires de nom `.NOM`) de votre répertoire initial
- Affichez exclusivement tous les fichiers et répertoires cachés (fichiers et répertoires de nom `.NOM`) de tous vos répertoires (i.e. de votre répertoire initial et récursivement de tous ses sous-répertoires)
- Comptez le nombre total de fichiers sources C (fichiers de nom `NOM.c`) présents dans toute l'arborescence auxquels vous avez accès en lecture
- Créez plusieurs fichiers avec l'extension `.bak` (des fichiers de nom `NOM.bak`) un peu partout dans votre arborescence simplement en copiant des fichiers existant, puis effacez-les en une seule commande
- Créez fichiers vides dans plusieurs de vos répertoires, puis effacez-les à l'aide d'une seule commande.

6 Gestion des processus

Sous Unix, un processus est une commande, ou plus généralement, un programme utilisateur quelconque. Deux exécutions d'un même programme engendrent 2 processus distincts. Chaque processus est identifié de façon unique par un numéro appelé PID (*process identifier*).
À tout moment, les programmes des utilisateurs du système s'exécutent simultanément. En fait, le terme simultanément est un abus de langage, car très souvent, les ordinateurs ne possèdent qu'un seul processeur, et un seul processus à la fois y accède selon un mécanisme de temps partagé (géré par un *scheduler*).
Tous les processus que vous aviez l'habitude de lancer jusqu'à présent étaient lancés en avant-plan. Il fallait donc attendre la fin de leur exécution pour pouvoir reprendre la main dans le terminal, et pouvoir en lancer un autre. Afin de remédier à cette attente qui peut ne pas être souhaitée, il y a la possibilité de lancer un processus en arrière plan. Pour cela il suffit de faire suivre la commande habituelle d'un `&`.

6.1 Quelques manipulations de processus

Les commandes **ps**, **kill**, **jobs**, **fg**, **bg** vous seront utiles pour les question suivantes :

- Tapez la commande `xclock -update 1`. Pouvez-vous exécuter une nouvelle commande depuis le terminal dans lequel vous avez lancé ce processus ?
- Suspendez l'exécution de ce processus sans le tuer. Qu'indique la commande `jobs` ? Reprenez son exécution en arrière plan.
- Lancez une nouvelle commande `xclock -digital -update 1` dans le même terminal. Interrompez-la. Qu'indique la commande `jobs` ? Faites passer le processus en avant plan puis en arrière plan.
- Que provoque la commande `kill -9 %1` ?
- Trouvez les PIDs des commandes que vous avez lancées.
- Affichez le(s) PID(s) d'une commande par son nom de manière que seul le PID et la commande exécutée s'affichent sur la ligne (ou sur chaque ligne s'il la même commande a été lancée plusieurs fois).

6.2 Lancements de processus différés

Les commandes **sleep**, **at**, **nohup** et **crontab** permettent de lancer un processus en différé.

- Trouvez deux manières de lancer la commande `xlogo` dans soixante secondes à partir de maintenant.
- Lancez le programme `xcalc` en tâche de fond à partir d'un nouvel `xterm`. A partir ce même `xterm`, lancez le programme `xlogo` avec la commande `nohup`. Détruisez le `xterm`. Déduisez-en le rôle de la commande `nohup`.
- Utilisez la commande **crontab** pour afficher le logo *X* toutes les minutes.

6.3 Observation de la charge du système

L'utilisation des commandes **top**, **time** et **uptime** vous permet d'avoir une idée de la charge ou de l'état du système. Essayez également la commande **KSysGuard** sous KDE (menu *système*).

- Regardez en direct la liste des processus et la consommation des ressources du système.
- Regardez la charge de votre machine.
- Mesurez le temps que met un `ls -R` depuis votre répertoire de travail.
- Mesurez le temps que met un `ls -R` depuis la racine de l'arborescence mais sans rien afficher.
- Depuis combien de temps votre machine tourne-t-elle sous Unix ?

7 Rudiments de programmation en shell

7.1 Fichiers de démarrage zsh

Le shell **zsh** (votre shell par défaut), comme tous les shells, utilise un certain nombre de fichiers de démarrage situés dans votre répertoire initial :

- `.zshenv` : ce fichier est lu à chaque invocation de **zsh**. Il doit contenir **uniquement** l'initialisation et l'exportation de variables d'environnement.
- `.zshrc` : ce fichier est lu à chaque invocation d'un **zsh** interactif et doit servir à définir les alias et les options du shell , ainsi que d'éventuelles fonctions ou redéfinitions de touche du clavier.
- `.zlogin` : exécuté uniquement au début d'un shell de login (généralement lancé une seule fois au début du TD), il ne dois **pas** contenir des commandes du type de celles présentes dans les fichiers précédents. Ce fichier est lu *après* le fichier `.zshrc`.
- `.zlogout` : exécuté uniquement à la fin d'un shell de login
- `.zprofile` : l'alternative au `.zlogin`, mais il est lu *avant* le fichier `.zshrc`. Vous ne devez **pas** utiliser à la fois un `.zlogin` et un `.zprofile`.

Visualisez vos fichiers de démarrage, et utilisez la commande *echo* pour vérifier quand et dans quel ordre ces fichiers sont lus par **zsh**.

7.2 Exercices

Pour réaliser les exercices qui suivent, considérez les commandes **for**, **seq**, **basename** et **finger**.

- Créez le répertoire *java* et sous ce répertoire, créez plusieurs fichiers dont les noms sont de la forme `XXtpN.java` en faisant varier `XX` (des caractères quelconques) et `N` (un entier entre 1 et 20). Des noms possibles sont `exo1tp1.java`, `intro-tp2.java`, `hello_tp2.java` ou encore `prog2.tp3.java`. Le contenu des fichiers importe peu et vous pouvez les créer avec la commande **touch** (le fichier créé est vide).
- Renommez chacun des fichiers de nom *NOM.java* en *NOM.java.bak*.
- Renommez chacun des fichiers de nom *NOM.java.bak* en *NOM.txt*.
- Créez dans le répertoire *java* les 20 sous-répertoires *TP1*, *TP2*,...*TP20*, puis déplacez tout fichier de nom *XXtp**i**.txt* dans le répertoire *java/TP**i***
- Créez le répertoire *users* et créez sous ce répertoire, autant de sous-répertoires qu'il y a d'utilisateurs au département SI, de façon que le nom de chaque sous-répertoire soit le *login* de l'utilisateur correspondant. Un utilisateur du département est quelqu'un dont le répertoire initial est sous le répertoire */u/essi*.
- Créez dans chacun des sous-répertoires du répertoire *users* un fichier *info.txt* réduit à une unique ligne contenant successivement l'UID, le prénom et le nom de l'utilisateur correspondant.
- Ajoutez dans chacun des fichiers *users/un_login/info.txt* les informations relative à l'utilisateur de login *un_login* affichées par la commande **finger**.
- Renommez tous les fichiers *users/un_login/info.txt* en *users/un_login.txt* et effacez tous les sous-répertoires de nom *users/un_login/*