Cours Systèmes
IUTA - Université Lyon 1

Systèmes d'exploitations - UNIX

Jean-Philippe Farrugia - Claude Ratard - Ariane Baron

jean-philippe.farrugia@iuta.univ-lyon1.fr

Consignes

- Cours/TP Système UNIX : 8 séances de 2 heures ;
- Principalement : utilisation UNIX ;
- Contrôle continu : Rapport à rendre.
  - Sujets sur Public/jpfarrug/ASR3SE/Linux.
  - Un rapport par sujet.
- Devoir : sur papier.

Généralités

- Un ordinateur contient :
  - Un ou plusieurs processeurs ;
  - Une mémoire ;
  - Des horloges ;
  - Des disques ;
  - Des périphériques d'entrée-sortie.
  - … => complexe et varié.
- Un système d'exploitation :
  - Est une interface entre l'utilisateur et le matériel ;
  - Assure l'équité de l'accès aux ressources entre tous les utilisateurs et pour toutes les applications disponibles ;
  - Fournit la base sur laquelle seront construits tous les programmes.

Généralités(2)

- Un système d'exploitation masque les éléments fastidieux liés au matériel ;
  - Ex pour READ : 13 paramètres sur 9 octets. En retour, le contrôleur envoie 23 champs d'états regroupés sur 7 octets.
- Un système d'exploitation gère les accès simultanés à une même ressource.
  - Ex : Comment gérer trois applications essaient d'imprimer en même temps ?

Généralités(3)

- **Différents types de systèmes d'exploitation :**
  - **Mono/multi-taches :**
    - Capacité du système à pouvoir gérer plusieurs processus simultanément, ou au moins à donner l'illusion que plusieurs taches s'exécutent en même temps.
    - Exemple : Windows 95, UNIX, OS2 sont multi-taches.
  - **Mono/multi-utilisateurs :**
    - Capacité du système à pouvoir gérer plusieurs utilisateurs utilisant les mêmes ressources matérielles ;
    - Pose le problème des droits d'accès ;
    - Exemple : UNIX, Windows XP sont multi-utilisateurs ;

Généralités(4)

- **Structure d'un système d'exploitation :**
  - **Un système moderne minimal comprend :**
    - Une unité qui gère de manière élémentaire les processus, les ressources(mémoires, périphériques…) et les fichiers. Cette unité est généralement appelée noyau ;
    - Une interface entre le noyau et les périphérique, gérée par un ensemble de gestionnaires de périphériques (ie « pilotes » ou « drivers ») ;
    - Une interface entre le noyau et les programmes utilisateurs, gérée par un ensemble d'appels systèmes ;

Plan

- Généralités
- Premier contact avec UNIX
- Le système de Fichiers
- Les expressions régulières et les filtres
- Les processus
- Les scripts
- Sed et Awk

UNIX ?

- Système d'exploitation de référence ;
- Un des plus anciens ;
- Aujourd'hui : plus un standard qu'un système proprement dit ;
- Version la plus connue du « grand public » : Linux
  - Unix libre ;
  - De plus en plus présent dans l'industrie et chez les particuliers.

UNIX(2)

- **Les bases d'UNIX :**
  - **« UNiplexed Information and Computing Service »;**
  - **Système multi-taches, multi-utilisateurs ;**
  - **Système de fichier à arborescence unique ;**
  - **Entrées-sorties compatibles fichiers, périphériques et processus ;**
  - **Quatre concepts élémentaires :**
    - Fichiers ;
    - Processus ;
    - Communication inter processus ;
    - Droits d'accès ;

UNIX(3)

- **Fichiers :**
  - Unité élémentaire de gestion de ressources ;
  - Non typé : l'extension a un but principalement informatif ;
  - Peut représenter plusieurs type de ressources : stockage d'information, périphérique, IPC…
  - Inclus dans un système de gestion de fichiers ;
- **Processus :**
  - Unité élémentaire de gestion des traitements ;
  - Comprends un espace d'adressage et supporte un ou plusieurs flot(s) d'exécution de programme (threads) ;
  - Chaque thread a sa pile et son propre contexte d'exécution ;

UNIX(4)

- **Communication inter-processus :**
  - **Permet d'arbitrer l'accès aux ressources entre plusieurs processus ou threads :**
    - Exemples : sémaphores, mutex…
  - **Permet le contrôle d'un processus par un autre ou par le noyau :**
    - Signaux…
  - **Permet la communication entre deux processus :**
    - Tubes et Sockets ;
- **Contrôle d'accès aux ressources :**
  - **Niveau logiciel :**
    - Chaque ressource admet un identificateur, un propriétaire et un ensemble de droits d'accès(lecture, écriture, exécution) répartis en 3 groupes (propriétaire, groupe du propriétaire, les autres) ;`
  - **Niveau matériel : mode superviseur et mode utilisateur ;**

UNIX(5)

- **Quelques bases pour débuter :**
  - **Première chose a faire sous UNIX : ouvrir une session**
    - À l'invite : entrer son login et son mot de passe ;
  - **Interface de commande principale : l'émulateur de terminal (Xterm) ;**
  - **Cet émulateur de terminal comprend un interpréteur de commandes ;**
  - **Une commande = une instruction permettant de donner des ordres au système.**

UNIX(6)

- **Une commande est toujours de la forme :**
  - **Commande options arguments**
    - Options : précisent le fonctionnement de la commande ;
    - Arguments : éléments que la commande doit prendre en compte et/ou traiter.
- **Une ligne de commande peut comprendre plusieurs commandes séparées par « ; » ;**
- **Exemple de commandes :**
  - **wc -l fichier.txt : permet de compter le nombre de lignes du fichier "fichier.txt";**
  - **man wc : permet d'obtenir de l'aide sur la commande wc ;**
- **Il est possible de créer un alias pour une commande :**
  - **Exemple : alias ll "ls -l" permet d'utiliser la commande ls -l en tapant uniquement ll.**

UNIX(7)

- Les premières commandes essentielles :
  - man : permet d'obtenir la documentation ;
  - cd : permet de changer de répertoire ;
  - pwd : affiche l'emplacement courant ;
  - ls : affiche le contenu d'un répertoire ;
  - echo : affiche un message ;
  - mkdir : crée un répertoire ;
  - id : affiche l'identité de l'utilisateur ;
  - cp : permet de copier un fichier ;
  - rm : efface un fichier ;
  - cat : affiche un fichier texte.

UNIX(8)

- **Exercice 1 :**
  - **Ouvrez une session et un terminal ;**
  - **Répertoires (cd, mkdir, ls) :**
    - Sous votre répertoire d'accueil, créez un répertoire SELINUX ;
    - Dans SELINUX, vous créerez un répertoire TPXX par séance ;
      - Aujourd'hui, il vous faut donc créer le répertoire TP01.
    - Vérifiez que les répertoires ont bien été créés.
  - **Fichiers (cp, cat, rm) :**
    - Dans le répertoire TP01, placez une copie du fichier /etc/passwd
    - Affichez le contenu de ce fichier ;
      - Que contient-il ?
    - Effacez cette copie.
  - **Autres (echo, id, date) :**
    - Affichez "bonjour" à l'écran ;
    - Testez les commandes "id" et "date" ;

UNIX(9)

- **Trucs et astuces :**
  - **“!i” :**
    - Exécute la “ieme” commande depuis l’ouverture du terminal ;
    - “history” permet de visualiser l’historique des commandes ;
    - Permet également de relancer une commande en spécifiant uniquement les premières lettres :
      - Exemple : !c relancera la dernière commande commençant par “c”.
  - **“$i” :**
    - Récupère le “ieme” dernier argument ;
    - Exemple : éxécutez la séquence de commandes suivante :
      - ls /etc/passwd
      - more $1

UNIX(10)

- **(Dé)Compression et (dés)archivage :**
  - **Compression : commande “gzip”**
    - Utilisation : gzip -n fichier_a_compresser
    - “n” représente le niveau de compression
      - n grand -> Très compressé, mais compression lente ;
      - n petit -> Inverse.
    - Par défaut, le fichier aura l’extension “.gz”
    - Exemple : copiez le fichier /etc/passwd et compressez le au niveau 4 ;
    - Pour décompresser : gzip -d fichier.gz .
  - **Archivage : commande “tar”**
    - Utilisation : tar -cf Nom_archive.tar Dossier_a_archiver
    - Désarchiver : tar -xf Nom_archive.tar ;
    - Note : en rajoutant l’option “z”, l’archive est compressée ;
    - Exemple : Créez, dans votre répertoire courant, une archive compressée de votre répertoire SELINUX.

UNIX(11)

- En cas d'erreur, vérifiez que :
  - La commande existe ;
  - Vous avez le droit d'exécuter la commande ;
  - Les options ne sont pas erronées ;
  - Les arguments ne sont pas erronés ;
- Dans la plupart des cas, l'utilisation de man permet de résoudre le problème.

Plan

- Généralités
- Premier contact avec UNIX
- Le système de Fichiers
- Les expressions régulières et les filtres
- Les processus
- Les scripts
- Sed et Awk

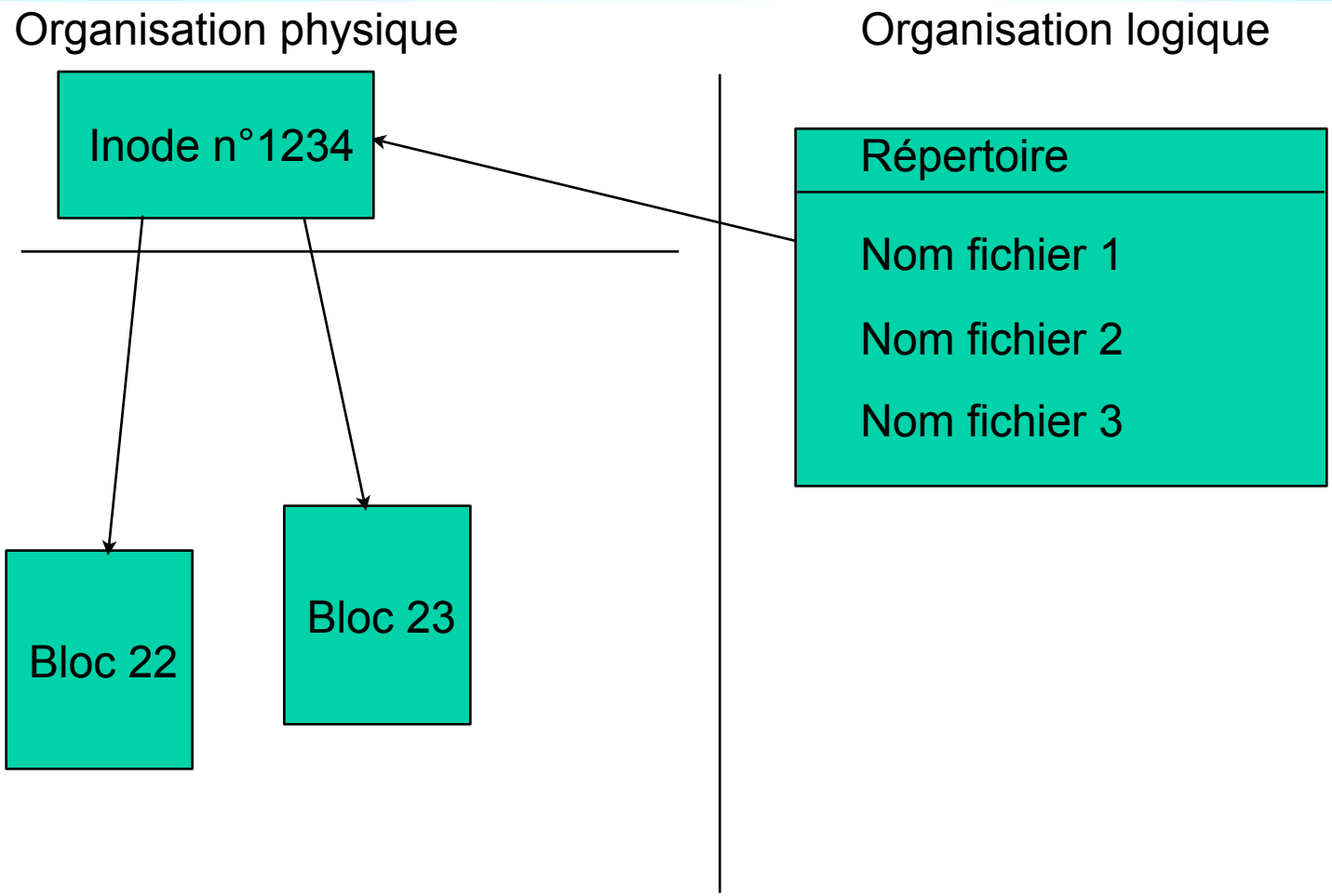
Le système de fichiers

- Sous UNIX, tout est fichier ;
- Plusieurs types :
  - Ordinaires ;
  - Catalogues ;
  - Liens ;
  - Spéciaux ;
  - Tubes et sockets ;
- Il existe une représentation hiérarchique du stockage de fichiers ;
- En interne (noyau), les fichiers ont tous la même structure.

Le système de fichiers(2)

- **Un fichier est une suite non structurée d'octets ;**
- **La structuration se fait au niveau de l'application ;**
- **Gestion :**
  - **Les données d'un fichier sont stockées dans un ou plusieurs bloc de données ;**
  - **Pour assurer la correspondance entre les blocs de données et les fichiers, le noyau gère une table ;**
  - **Un fichier est repéré par un enregistrement de cette table : son inode ;**
    - l'inode contient les informations associées au fichier (droits d'accès, propriétaire, type...) en plus de son adresse physique.
- **Il y a un système de gestion de fichiers par zone de stockage matérielle ;**

Le système de fichiers(2)



Le système de fichiers(3)

- **Fichier ordinaire ou régulier :**
  - **C'est un fichier qui n'est ni un catalogue, ni un lien, ni un tube, ni une socket, ni un fichier spécial ;**
  - **Un fichier est généralement repéré par son nom ;**
  - **Les fichiers dont le nom commence par « . » sont des fichiers cachés.**

Le système de fichiers(4)

- **Fichier catalogue ou répertoire :**
  - **C'est un fichier qui contient une liste de fichiers ;**
  - **Toutes les versions d'UNIX ont une racine unique désignée par « / »(slash) ;**
  - **Tous les répertoires possèdent obligatoirement dans leur liste deux autres répertoires :**
    - « . » est un synonyme pour le répertoire lui même ;
    - « .. » est un synonyme pour le répertoire qui le contient ;
  - **Pour trouver un fichier ordinaire, il faut connaître son chemin d'accès.**

Le système de fichiers(5)

- Un chemin = un point de départ + une liste des répertoires à traverser pour arriver au répertoire destination ;
- La liste de répertoire est composée de répertoires séparés par le symbole “ / “ ;
  - Si le point de départ est la racine, le chemin est dit absolu ;
  - Dans le cas contraire, c’est un chemin relatif ;
  - La racine est représentée par “ / ”.
- Astuce : “~” représente le répertoire d’accueil de l’utilisateur courant.

Le système de fichiers(6)

- Hierarchie standard UNIX :
  - /bin : commandes principales ;
  - /dev : fichiers de périphériques ;
  - /etc : fichiers de configuration spécifiques à la machine ;
  - /home : répertoire des utilisateurs ;
  - /lib : librairies partagées ;
  - /sbin : commandes d'administration essentielles ;
  - /tmp : fichiers temporaires ;
  - /usr : seconde hiérarchie ;
  - /var : données variables ;

Le système de fichiers(7)

- Seconde hierarchie :
  - /usr/X11R6 : X window (interface graphique) ;
  - /usr/bin : commandes utilisateur ;
  - /usr/include : entètes pour les fichiers C ;
  - /usr/lib : bibliothèques utilisateur ;
  - /usr/local : hiérarchie locale ;
  - /usr/sbin : Commande d'administration non vitales ;
  - /usr/share : données indépendantes de l'architecture ;
  - /usr/src : code source ;

Commandes associées

- **Fichiers ordinaires :**
  - **cat : affiche le contenu du fichier ;**
  - **stat : affiche les caractéristiques d'un fichier ;**
  - **ls : affiche les caractéristiques d'une liste de fichier ;**
  - **rm nom_fichier: supprime un fichier ;**
  - **touch : permet de modifier les caractéristiques de date d'un fichier, voire de créer un fichier vide.**
  - **cp : permet de copier un fichier ;**
  - **mv : permet de déplacer ou de renommer un fichier ;**
  - **more : alternative pour visualiser le contenu d'un fichier**

Commandes associées(2)

- **Répertoires :**
  - **pwd : donne le chemin absolu du répertoire courant ;**
  - **cd : permet de changer le repertoire courant ;**
  - **ls : permet d'obtenir la liste des fichiers contenus dans un répertoire ;**
  - **mkdir : permet de créer un répertoire ;**
- **Alias :**
  - **La commande alias permet de créer un alias...**

Travail Pratique

- **Exercice 2** : manipulations de fichiers
  - Placez vous dans le répertoire /etc et copiez le fichier « fstab » dans votre répertoire TP01 ;
  - Revenez dans TP01 et renommez « fstab » en « table » ;
  - Créez un répertoire « systeme » ;
  - Déplacez « table » dans « systeme » ;
  - Sans vous déplacer, faites une copie de « table » nommée « table1 » dans votre répertoire courant ;
  - Affichez le contenu du répertoire courant ;
  - Affichez l'inode de table1.
  - Renommez « table1 » en « table2 » ;
  - Copiez « table2 » dans le répertoire « TP1 » en le renommant « table3 » ;

Le système de fichiers(8)

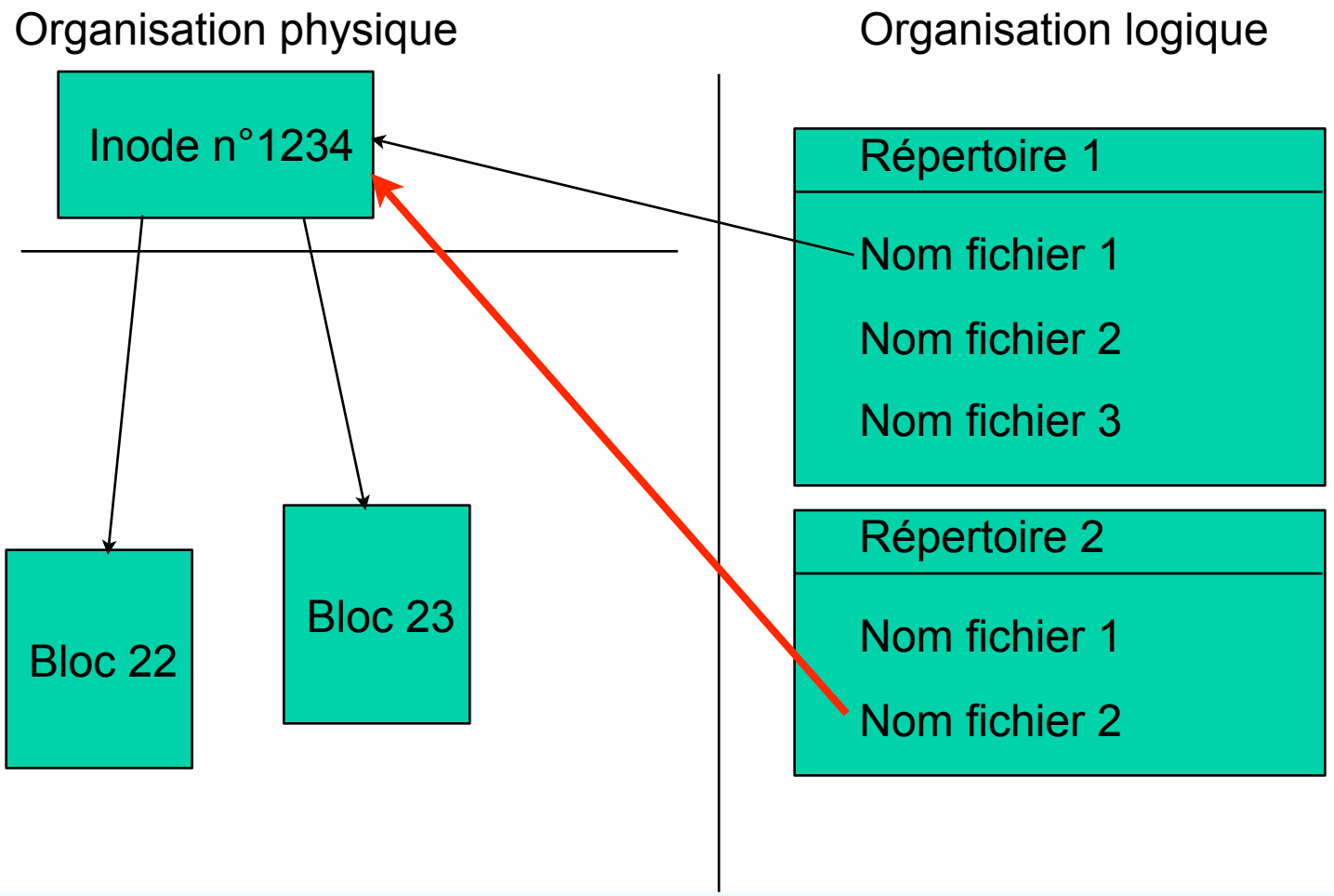
- **Fichiers spéciaux :**
  - **Un fichier spécial représente généralement un périphérique ;**
    - Un périphérique = matériel physique connecté à l'unité centrale : disque, souris, réseau...
    - Sous linux, un périphérique est présent dans l'arborescence, sous le répertoire /dev ;
    - Un pilote est une fonction du système permettant d'exploiter le périphérique via les appels systèmes ;
  - **Un périphérique peut être virtuel :**
    - Est géré comme un périphérique "normal", mais ne correspond à aucune réalité physique.
    - Exemples : Ecrans virtuels, partitions logiques, poubelle...

Le système de fichier(9)

- **Fichiers liens : deux types**
  - **Un lien physique est l'entité qui relie un nom de fichier à l'inode correspondant ;**
    - Plusieurs liens physiques pour le même inode = une même donnée est repérée par plusieurs noms de fichiers différents ;
    - N'a pas d'équivalent sous Windows.
  - **Un lien symbolique est un fichier qui ne contient que le chemin et le nom d'un autre fichier ;**
    - l'inode pointé par le lien symbolique renvoie sur un nom de fichier au lieu d'un bloc de données ;
    - En langage Windows, un lien symbolique est un raccourci ;
    - En manipulant le lien, on manipule en fait le fichier dont le chemin est stocké dans le lien ;
  - **Commande associée : ln**
    - Sans option : lien physique ;
    - Avec l'option -s : lien symbolique ;

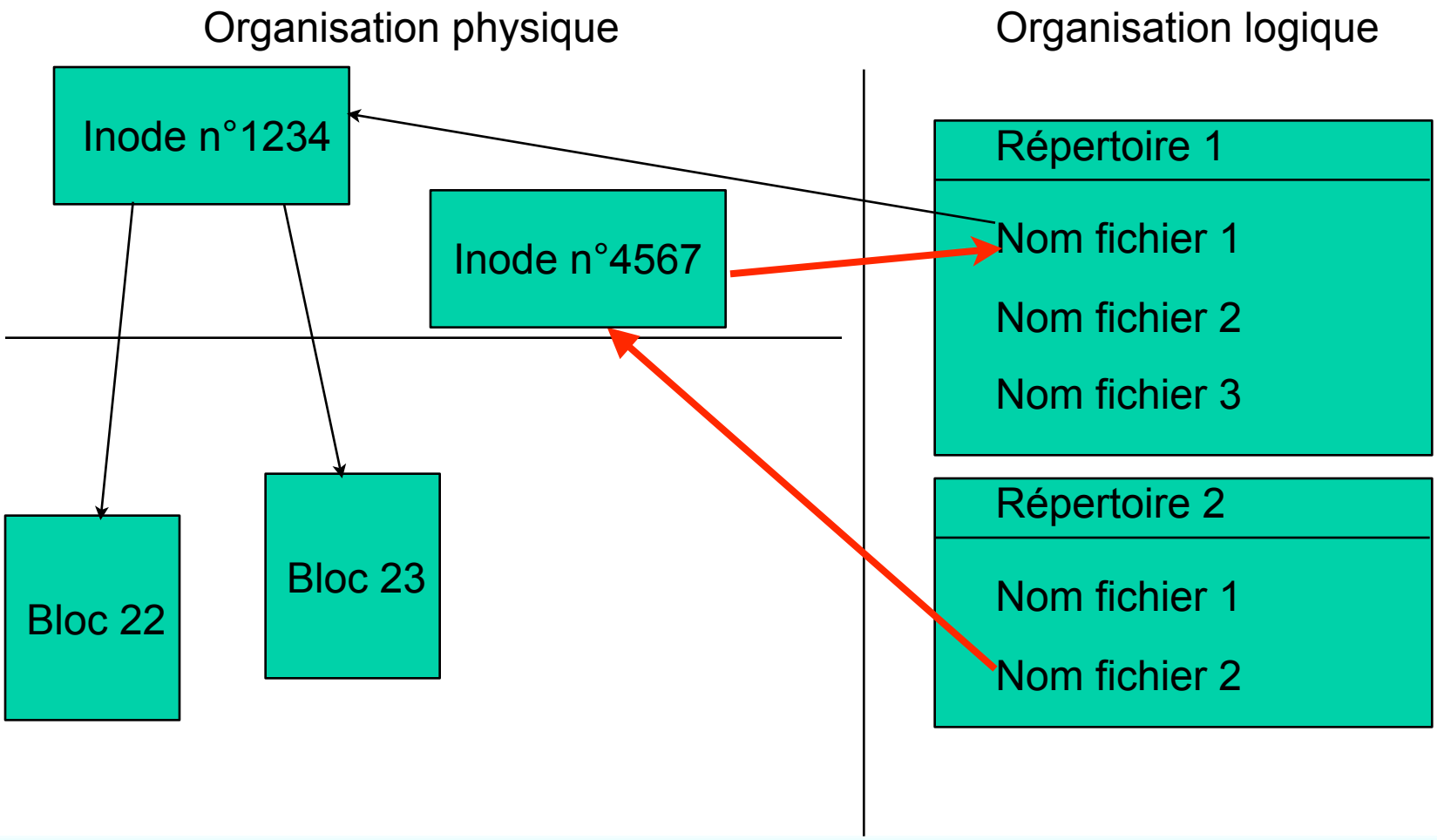
Le système de fichiers(9)

Lien physique



Le système de fichiers(9)

Lien symbolique



Le système de fichier(10)

- **Tubes**
  - **Un tube est un fichier permettant à deux processus locaux de communiquer ;**
  - **Dans une commande, un tube permet de rediriger la sortie standard d'une première commande vers l'entrée standard d'une deuxième.**
    - Format : commande1 | commande 2
    - Exemple : essayez de rediriger la sortie standard de "ls" vers l'entrée standard de "more".
- **Sockets :**
  - **Une socket est un tube entre deux processus distants ;**
  - **Caractérisée par un numéro de port et une adresse IP ;**

Travail Pratique

- **Exercice 3 : les liens**
  - Faites une copie nommée “cp_passwd” du fichier /etc/passwd dans votre répertoire d’accueil. Quel est l’inode de cette copie ? Comparez-le avec l’inode de /etc/passwd ;
  - Créez un lien physique, nommé “ln_passwd”, de cp_passwd au même endroit.
  - Modifiez le fichier cp_passwd, par exemple en ajoutant une ligne quelconque en haut du fichier ;
  - Ouvrez le fichier ln_passwd. Est il modifié ? Pourquoi ? Quel est son inode ?
  - Supprimez le fichier cp_passwd. Le fichier ln_passwd est il supprimé ? Pouvez vous lire son contenu ?
  - Recommencez les étapes 2 à 5 en remplaçant le lien physique par un lien symbolique. Commentez les

Le système de fichiers(10)

- UNIX est multi-utilisateurs :
  - Plusieurs utilisateurs peuvent utiliser simultanément la machine ;
  - => Les fichiers sont propres à chaque utilisateur ;
- Chaque utilisateur a :
  - Un login, identifié au niveau du noyau par un UID ;
  - Un groupe, identifié au niveau du noyau par un GID ;
  - Utilisateur particulier : root (ou super utilisateur).
- Dans la hiérarchie :
  - Login et uid sont dans le fichier /etc/passwd
  - Groupe et gid sont dans le fichier /etc/group

Le système de fichiers(11)

- **Gestion des droits :**
  - **Pour gérer l'aspect multi-utilisateur, les fichiers UNIX ont des droits :**
    - Lecture ;
    - Écriture ;
    - Exécution.
  - **Les droits sont différents pour :**
    - L'utilisateur ;
    - Le groupe ;
    - Les autres.
  - **Pour un répertoire : le droit d'exécution est un droit d'accès au répertoire;**
  - **Un fichier texte avec les droits d'exécution devient un script.**

Commandes associées

- **Gestion des droits et des utilisateurs :**
  - **chmod : modifier les droits d'un fichier ou d'un répertoire ;**
  - **Deux utilisations possibles :**
    - chmod utilisateur operation droits_d'accès fichier
      - Utilisateur : u, g, o
      - Operation : +,-
      - Droits d'accès : r, w, x
      - Exemple : comment, pour un fichier, changer les droits du propriétaire et du groupe en lecture seulement ?
        - Réponse : chmod ug+r-wx fichier
    - chmod code_octal fichier
      - Le code octal est calculé sur 3 x 3 bits, représentant les droits d'accès pour le propriétaire, le groupe et les autres.
      - Pour l'exemple précédent :
        - chmod 440 fichier
  - **chown / chgrp : modifier le propriétaire/groupe d'un fichier ;**
  - **Utilisation :**
    - chown nouveau_propriétaire fichier
    - chgrp nouveau_groupe fichier

Travail Pratique

- **Exercice 4 : Droits d'accès**
  - **Dans votre répertoire racine, creez un fichier « private.txt » dont la lecture et l'écriture ne sont autorisées que pour son propriétaire ;**
  - **Tapez ls –l private.txt. Quel est le résultat ?**
  - **Modifiez de nouveau les droits :**
    - Ajoutez au groupe la possibilité de lire le fichier en utilisant la forme classique de chmod ;
    - Faites de même avec les autres utilisateurs, mais avec la forme octale ;
    - Comment vérifiez-vous le bon fonctionnement de votre commande ?
  - **Modifiez les droits pour tout interdire a tout le monde…**
    - Essayez de lire le fichier. Que se passe t il ?
    - Rétablissez les droits de lecture pour le propriétaire ;
    - Essayez de modifier le contenu du fichier. Que se passe t il ?
    - Rétablissez les droits d'écriture pour le propriétaire.

Travail Pratique

- **Exercice 4 : Droits d'accès, suite**
  - **Créez un répertoire droit_rep dans votre répertoire racine et un fichier droit dans ce répertoire ;**
  - **Nous allons traiter chacun des cas suivants pour le répertoire droit_rep :**
    - rw-, ---, r--, --x
  - **Après exécution de chacune des commandes correspondantes :**
    - Essayez d'aller dans le répertoire ;
    - Essayez de lire le contenu du répertoire ;
    - Essayez de modifier le fichier droit.
  - **Donnez le fichier "droit" à votre voisin de gauche.**
    - Comment faites vous pour vérifier que cela a fonctionné ?
  - **A quoi sert la commande umask ?**

Le système de fichiers(12)

- **Descripteurs - entrée et sortie standard**
  - **Un accès à une entrée/sortie quelconque est nommé descripteur ;**
  - **Sous UNIX, un descripteur est un fichier ;**
  - **Descripteurs particuliers : entrée et sortie standard :**
    - Fournis par le système ;
    - Permettent de lire des données au clavier et d'en écrire à l'écran ;
  - **Trois descripteurs à connaître sous UNIX :**
    - Entrée standard : stdin
    - Sortie standard : stdout
    - Sortie erreur : stderr
  - **L'entrée et la sortie standard peuvent être redirigées.**

Le système de fichiers(13)

- **Redirections :**
  - **Les entrées / sorties peuvent être redirigées avec les symboles “<” et “>” ;**
  - **“>” sert à rediriger la sortie standard ;**
    - Exemple : ls > log_ls.txt redirige la sortie de ls dans le fichier log_ls.txt
  - **“<” sert a rediriger l’entrée standard ;**
    - Autre exemple : read < read_data.txt redirige l’entrée standard de read vers le fichier read_data.txt
  - **“>&” sert a rediriger à la fois la sortie standard et la sortie erreur ;**
  - **“>>” et “<<” permettent d’ajouter à la fin d’un fichier existant au lieu de l’écraser.**

Travail Pratique

- **Exercice 5 : redirections et périphériques**
  - **Stockez, dans un fichier texte, la liste des fichiers et répertoires de votre répertoire d'accueil ;**
    - Vérifiez que cela a fonctionné ;
  - **Ecrivez "fin" à la fin de ce fichier ;**
    - il est évidemment interdit d'utiliser un éditeur de texte...
  - **Essayez de stocker le man de ls ;**
  - **Les périphériques :**
    - Redirigez la sortie de ls dans /dev/null. Que se passe-t-il ? Pourquoi ?
    - Essayez la commande tty. Quelle est son utilité ?
    - Ouvrez un deuxième terminal ;
    - Ecrivez dans le deuxième terminal depuis le premier. Quelle est la commande utilisée ? Expliquez.

Plan

- Généralités
- Premier contact avec UNIX
- Le système de Fichiers
- Les expressions régulières et les filtres
- Les processus
- Les scripts
- Sed et Awk

Les expressions régulières

- Sous UNIX, on peut utiliser des méta-caractères pour décrire un ensemble de caractères :
  - ? désigne un caractère quelconque ;
  - * désigne de 0 à n caractères ;
  - [xyz] désigne x OU y OU z ;
  - [a-g] désigne un caractère entre a et g ;
  - [^xyz] désigne un caractère autre que x, y et z;
  - ^ désigne le début de ligne, $ la fin de ligne.
- Ces caractères peuvent être utilisés au sein d'expressions régulières :
  - x* désigne zéro ou plusieurs occurrences du caractère "x" ;
  - x+ désigne une ou plusieurs occurrences du caractère "x" ;
  - x? désigne une occurrence unique du caractère "x".

Les expressions régulières (2)

- **Remarque : dans une expression régulière, “.” remplace “?” pour désigner un caractère quelconque.**
- **Exemples :**
  - **En vous servant de ls avec des méta-caractères, listez les fichiers de votre répertoire d’accueil ...**
    - ... qui commencent par a avec un z en troisième lettre ;
      - Créez en un ou plusieurs au besoin...
    - ... qui commencent par une majuscule ;
    - ... dont le nom fait 3 lettres ;
    - ... dont le nom comprend un chiffre ;
- **Les méta-caractères sont très utiles avec les filtres.**
- **Les expressions régulières sont souvent utilisées avec la commande grep.**

Les filtres

- Un filtre est une commande permettant de modifier ou de sélectionner tout ou une partie d'un flux texte :
  - Flux = sortie d'un autre processus, redirection de l'entrée standard ou fichier.
- Exemples de filtres :
  - tr : remplace un caractère par un autre ;
  - grep : sélectionne une partie d'un flux en fonction de critères précis ;
  - sort : permet de trier un flux ;
  - find : permet de localiser un/des fichier ;
  - cut, diff, paste, uniq...

Les filtres(2)

- **grep :**
  - **Recherche une chaîne de caractère dans un flux texte.**
  - **Grep est très souvent utilisé avec le pipe "|" :**
    - Permet la redirection de la sortie standard d'un processus sur l'entrée standard d'un autre.
  - **<u>Exemple</u> : essayez d'utiliser grep pour n'afficher, dans votre répertoire d'accueil, que les fichiers et répertoires dont le nom contient la lettre "a" ;**
    - Est il possible de faire sans grep ?
  - **Commande très usitée...**

Les filtres(3)

- **sort :**
  - **Permet de trier les lignes d'un flux selon un champ spécifié ;**
  - **Le séparateur par défaut, pour les champs, est l'espace ;**
  - **"man sort" donne les détails du fonctionnement ;**
  - **<u>Exemple</u> : Triez le contenu de votre répertoire courant :**
    - Par ordre alphabétique ;
    - Par mois de création ;
    - Par taille en octets ;
    - Par droits d'accès ;

Les filtres(4)

- **Find :**
  - **Permet de retrouver un fichier dans une arborescence donnée.**
  - **Fonctionnement : find racine expression comm :**
    - “racine” : racine de l’arborescence à explorer ;
    - “expression” : motif et critères à rechercher ;
    - “comm” : commande à éxecuter (facultative).
  - **Consulter man find**
  - **Exercice : que font les commandes suivantes ?**
    - find ./ -name ‘*.c’ -size 80
    - find / -user $USER

Les filtres(5)

- tr : substitue un caractère par un autre
  - Ex : utilisez tr pour remplacer, dans le fichier '/etc/passwd', les ':' par ' '.
- cut : permet de couper les colonnes d'un flux et d'en sélectionner certain champs seulement.
  - Ex : comment extraire le premier et le troisième champ du fichier /etc/passwd ?
- paste : permet de "coller" deux flux :
  - Tester la commande paste avec un fichier "noms" et un fichier "prénoms", que vous réunirez en un seul flux.
- diff : trouve les différences entre deux fichiers.

Travail Pratique

- **Exercice 6 : les filtres**
- **Constituer un annuaire de quelques correspondants avec le format suivant : nom:ville:numéro de téléphone**
- **Répondre aux questions suivantes :**
  - **Visualisez uniquement les noms et numéros de téléphones de l'annuaire ;**
  - **Visualisez les informations sur un correspondant connu par son nom ;**
  - **Visualisez l'annuaire trié sur le nom ;**
  - **Constituez un nouvel annuaire en enlevant un correspondant ;**
  - **Visualisez le nombre de correspondants ;**
  - **Visualisez les correspondants qui habitent Villeurbanne ;**
  - **Visualisez les correspondants dont le nom commence par "d" ;**

Travail Pratique

- **Exercice 6, suite : durée 15 minutes.**
  - **Créez un fichier “fruits.txt” contenant les mots suivant (un par ligne) : tomate, poire, pomme, cerise, Fraise, fraise, courgette, POMME3, afraise.**
  - **Affichez uniquement :**
    - Les lignes dont le nom est Fraise ou fraise ;
    - Les lignes dont le nom se finit en “se” ;
    - Les lignes ou “ai” est présent dans le nom ;
    - Les lignes dont le nom contient un chiffre ;
    - Les lignes dont le nom fait exactement 5 lettres ;
  - **Remarque : dans une expression régulière, “.” remplace “?” dans la substitution d’un caractère.**

Travail Pratique

- **Exercice 6 : les filtres, suite.**
  - **Chercher dans votre arborescence les fichiers :**
    - Dont l'extension est ".txt" ;
    - Commençants par "X" ou "x" ;
    - Dont les noms contiennent au moins un chiffre ;
  - **Chercher dans "~" les fichiers dont la taille dépasse un mégaoctet ;**
  - **Combien y a t il de fichiers dont le nom a "u" pour troisième lettre et ayant les droits positionnés en "rwx --- ---" dans votre arborescence ?**
    - Créez en un au besoin...
  - **Supprimez tous les fichiers "core" présents dans votre arborescence.**
    - Pour vérifier le bon fonctionnement, vous créerez deux fichiers nommés "core" : le premier sous votre répertoire d'accueil, le deuxième dans le répertoire TP01.
    - Les deux fichiers doivent bien entendus être effacés avec une seule commande...

Plan

- Généralités
- Premier contact avec UNIX
- Le système de Fichiers
- Les expressions régulières et les filtres
- Les processus
- Les scripts
- Sed et Awk

Processus

- **Présentation**
  - **Un processus est un programme binaire en cours d'exécution ;**
  - **L'espace de mémoire d'un processus est composé de trois segments :**
    - Du code à exécuter ;
    - Une zone de données ;
    - Une pile d'exécution.
  - **Le système maintient une table pour gérer l'ensemble des processus ;**
  - **Un processus s'exécute dans un contexte d'exécution ;**

Processus(2)

- Le contexte d'exécution d'un processus contient :
  - Un numéro d'identification unique nommé pid ;
  - L'identification de l'utilisateur qui a lancé le processus (uid) ;
  - Le répertoire courant ;
  - Les fichiers ouverts par ce processus ;
  - La taille maximale des fichiers que ce processus peut créer ;
  - La priorité ;
  - Les temps d'exécution ;
  - Le terminal de contrôle ;
  - …
- Un processus renvoie toujours une valeur de statut quand il se termine :
  - Ce code est accessible en visualisant la variable « $? » avec la commande « echo » ;

Processus(3)

- **Hiérarchie de processus :**
  - **Presque tous les processus ont un processus père ;**
    - Le « père » est à l'origine de la création du « fils » ;
  - **Le pid du père est nommé ppid ;**
  - **Le fils hérite de l'environnement du père, PID excepté ;**
  - **Le processus « init » est le premier processus lancé sous UNIX ;**
    - Init est « l'ancêtre » de tous les processus utilisateurs ;
    - Init lance deux sortes de processus : des « démons » et des processus interactifs.
  - **Le processus père peut lancer un processus fils en mode synchrone ou asynchrone (tache de fond) ;**
    - Dans un terminal, le symbole « & » sert à lancer un processus en asynchrone.

Processus(4)

- **Commandes associées aux processus :**
  - **Pour lancer un processus en tache de fond : « & »**
  - **wait :**
    - Permet d'attendre la mort d'un processus lancé en tache de fond dont le pid est connu ;
  - **ps :**
    - Permet de visualiser une liste de processus lancés ;
    - Par défaut, ps visualise les processus lancés depuis le terminal auquel elle est associée ;
  - **nice :**
    - Permet de régler le niveau de priorité d'un processus à son lancement ;
  - **exec :**
    - Permet de substituer le programme d'un processus par un autre ;
    - Exemple : essayez la commande suivante :
      - exec ls
    - Que se passe-t-il ? Pourquoi ?

Travail pratique

- **Exercice 7 : les processus**
  - **A l'aide de la commande ps, affichez la liste de tous les processus tournant sur votre machine ;**
    - Que signifient les indications présentes ?
  - **Listez tous les processus vous appartenant tournant sur votre machine ;**
  - **Pouvez vous déterminer le nombre approximatif de processus lancés depuis le démarrage de la machine ?**
    - même question pour les processus tournant actuellement ?
  - **Donnez la liste des trois derniers « ancêtres » de la commande ps en cours d'exécution ;**
  - **Testez la commande « top ».**

Processus(5)

- **Commandes associées aux processus, suite :**
  - **Pour arreter un processus synchrone : « ctrl-c » ;**
  - **Pour suspendre un processus synchrone : « ctrl-z » ;**
    - Un processus suspendu est en attente. Il est susceptible d'être repris.
  - **jobs :**
    - Permet d'obtenir la liste des processus lancés en tache de fond.
  - **Pour faire passer un processus asynchrone en synchrone : fg ;**
  - **Pour faire passer un processus synchrone en asynchrone : bg ;**
    - Pour ces deux dernières commandes, le paramètre est le numéro de tache donné par « jobs ».
  - **Kill pid :**
    - Permet de « tuer » ou d'envoyer un signal (ordre) à un processus ;
    - kill -l permet de consulter la liste des signaux disponibles ;
    - Plus de détails en programmation système…

Travail pratique

- **Exercice 7, suite : avant-plan et arrière-plan**
  - **Créez un programme en c qui incrémente une variable i indéfiniment et qui affiche i lorsque sa valeur est un multiple de 100.**
    - Conseil : utilisez la commande « sleep » pour ralentir l'exécution et y voir quelque chose !
  - **Compilez votre programme avec cette commande :**
    - cc -o compteur votre_programme.c
  - **Lancez quatre instances du processus « compteur » en tache de fond ;**
  - **A l'aide des commandes jobs, bg et fg, mettez au premier plan la troisième, suspendez la, et relancez la en tache de fond ;**
  - **Arrêtez tous les compteurs avec ps et kill ;**
  - **Testez la commande killall pour tuer tous les compteurs ;**

Plan

- Généralités
- Premier contact avec UNIX
- Le système de Fichiers
- Les expressions régulières et les filtres
- Les processus
- Les scripts
- Sed et Awk

La programmation Shell

- **Le Shell ?**
  - **C'est le nom de l'interpréteur de commande ;**
  - **Principalement deux familles de Shell :**
    - Bourne Shell : syntaxe proche des premiers shells unix ;
      - /bin/sh
    - C Shell : syntaxe « proche » de celle du C ;
      - /bin/csh
  - **Sous Linux, il existe deux variations :**
    - Bourne Again Shell : Bourne Shell augmenté de toutes les fonctionnalités du C Shell ;
    - Tcsh : extension du C shell d'origine ;

La programmation Shell(2)

- **Il existe deux types de commandes :**
  - **Interne : une commande interne est une commande intégrée au Shell. Pour traiter une commande interne, le Shell effectue lui même un traitement spécifique, sans créer de processus.**
    - Essayez de trouver des commandes internes à l'aide de « which » ;
    - Selon le type de Shell, les commandes internes peuvent être différentes.
  - **Externe : une commande externe est un fichier exécutable ;**
    - L'emplacement de ces fichiers exécutables est spécifié dans une variable d'environnement nommée PATH ;
    - La commande echo permet de visualiser la variable PATH ;
      - echo $PATH
    - PATH est modifiable : vous pouvez créer vos propres commandes et y accéder depuis le Shell ;

La programmation Shell(3)

- **Le Shell permet la création de scripts :**
  - **Un script est un fichier texte exécutable ;**
  - **Il contient les commandes à exécuter ;**
  - **Attention : pour qu'un fichier texte soit exécutable, il faut lui donner des droits d'exécution avec chmod.**
- **Lorsqu'une commande est tapée dans le Shell, le comportement est le suivant :**
  - **Si la commande est interne, celle ci est exécutée au sein du Shell courant ;**
    - Dans ce cas, aucun processus n'est créé ;
  - **Si elle n'est pas interne, le Shell recherche dans les répertoires situés dans PATH pour trouver cette commande :**
    - Si c'est un binaire, il est exécuté ;
    - Si c'est un fichier texte (script), les commandes sont lues dans ce fichier et exécutées ;
    - Dans un cas comme dans l'autre, un/des nouveau processus est créé.

La programmation Shell(3)

- **Exercice 8 : un script simple**
- **Créez un script qui effectue à la suite les opérations suivantes :**
  - **Créer un répertoire “Essai_Script” ;**
  - **Créer un fichier vide “toto” dans ce répertoire ;**
  - **Créer une copie de “toto” nommée “toto2”, toujours dans ce répertoire ;**
  - **Stocke une liste des fichiers de votre répertoire d’accueil, toujours dans “Essai_Script” ;**
  - **Afficher “voila, c’est fait !”**
- **Testez le bon fonctionnement de votre script.**

La programmation Shell(4)

- **Variables :**
  - **Il est possible de créer et d'affecter une valeur à une variable dans un Shell ;**
  - **Une variable est désignée par son nom précédé du caractère '$' ;**
  - **Une variable Shell n'est pas typée : son contenu est une chaîne de caractères totalement quelconque ;**
  - **L'affectation d'une variable change selon le type de Shell :**
    - En bash : la commande "=" permet d'affecter une variable
      - test=valeur
    - Lors de l'affectation, le signe "$" est absent.
  - **Echo permet de visualiser une variable ;**
  - **Attention :**
    - Une variable est locale au Shell où elle a été affectée ;
    - Sur certains Shells, une variable non déclarée a une valeur nulle.

La programmation Shell(5)

- **Variables :**
  - **Pour bien délimiter une variable, il faut encadrer son nom entre accolades, $ exclu ;**
    - Exercice : Créez une variable test dont la valeur est « riendutout » ;
    - Testez les commandes suivantes :
      - echo $test
      - echo A$test
      - echo A$testA
      - echo A${test}A
    - Commentez.
  - **Une variable peut être utilisée dans une commande :**
    - Créez une variable « repertoire » dont la valeur est « /bin » ;
    - Utilisez la avec la commande ls ;

La programmation Shell(6)

- **Exercice :**
  - **Testez les commandes suivantes :**
    - A=10 ; B=A ; echo $B ;
    - B=$A ; echo $B ;
    - A=3 ; echo $B ;
  - **Commentez les résultats :**
    - Une variable est elle réévaluée à chaque utilisation de echo ?

La programmation Shell(7)

- **Variables :**
  - **Il est possible de récupérer le résultat d'une commande dans une variable avec le caractère ` ;**
  - **<u>Exercice</u> : Testez le résultat des commandes suivantes :**
    - datedujour=`date`
    - echo $datedujour
    - datedujour=la date du jour est `date`
    - datedujour=‘ la date du jour est `date` ’
    - echo $datedujour
    - datedujour= ” la date du jour est `date` ”
    - echo $datedujour
  - **Qu'en déduisez vous ?**

La programmation Shell(8)

- Il est possible d'obtenir la liste des variables actuellement définies avec la commande set ;
- Pour détruire une variable, il faut utiliser la commande unset ;
- Rappel : une variable est <u>locale à un shell</u>
  - Pour créer une variable disponible dans tous les shells, celle ci doit être créée à l'ouverture de session ;
  - Il faut pour cela modifier un fichier de configuration spécifique pour y placer la commande d'affectation

La programmation Shell(9)

- **Variables automatiques utiles :**
  - **Le shell possède un certain nombre de variables prédéfinies pour son fonctionnement**
    - Exemple : la variable $PATH donne le chemin d'accès par défaut aux exécutables ;
  - **Autres variables automatiques :**
    - $_ : la dernière commande ;
    - $? : le code de retour du dernier processus lancé ;
    - $! : le numéro du dernier processus lancé en arrière plan ;
    - $PPID : le numéro du processus père du processus courant ;
    - $PWD : le chemin absolu du répertoire courant ;

La programmation Shell (10)

- **Affectation interactive d'une variable :**
  - **La commande read permet de lire sur l'entrée standard la valeur à affecter à une variable ;**
    - Read a ;
    - 33 ;
    - Echo $a
  - **On peut affecter plusieurs variables à la fois :**
    - Read b c d
    - Trois mots simultanément
    - Echo $b $c $d
- **<u>Exercice</u> :**
  - **Réalisez un script qui affiche un message de bienvenue, qui lit une phrase de quatre mots au clavier et qui l'affiche à l'envers ;**

La programmation Shell (11)

- **Arguments - Paramètres**
  - Une commande exécutable, script ou autre, accepte des arguments ;
  - Pour un script, les arguments sont accessibles sous la forme de variables nommées $0, $1, $2 ;
  - $0 correspond à la commande elle même, $1 au premier argument, $2 au deuxième…
  - Si il y a plus de 9 arguments, il faut utiliser la segmentation avec les accolades : ${10} ;
- **Exercice :**
  - Refaites l'exercice précédent sans utiliser la commande read ;

La programmation Shell (12)

- **Structures de contrôle**
  - **Structure for :**
    - For nom in liste
    - Do
    - Commandes
    - Done
  - **La liste est une suite de valeur séparée pas des espaces ;**
    - Si aucune liste n'est fournie dans le code du script, celle ci est constituée par la liste des arguments ;
  - **Exercices :**
    - Ecrivez un script un compte à rebours qui compte de 5 à 1 et qui affiche ensuite « décollage » ;
    - Ecrivez un script qui crée 5 fichiers fic1.txt à fic5.txt et qui les copie dans un répertoire passé en paramètre ;
      - même question avec les fichier fic_a1.txt,..,fic_a5.txt, fic_b1.txt…fic_b5.txt… jusqu'à fic_e5.txt ;
    - Ecrivez un script qui change les droits à rwxr-- r-- des fichiers passés en paramètres.

La programmation Shell (13)

- **Structures de contrôle, suite :**
  - **Il est possible d'utiliser une variable comme liste pour la structure for ;**
  - **Il est possible, en utilisant « ` » d'utiliser le résultat d'une commande comme liste :**
  - **Exercice :**
    - Ecrivez un script qui ajoute un « a » au début du nom de tous les fichiers d'un répertoire passé en paramètre ;

La programmation Shell (14)

- **Structures de contrôle :**
  - **Structure if :**
    - If commande1
    - Then commande2
    - Else commande3
    - Fi
  - **La partie else est optionnelle ;**
  - **La commande 1 est évaluée par rapport à son code de retour ;**
- **En utilisant un fichier avec des noms utilisateur, écrivez un script « existuser » permettant de déterminer si le nom passé en paramètre à la commande correspond à un utilisateur présent dans le fichier.**

La programmation Shell (15)

- Il est possible d'imbriquer les if sous la forme suivante :
  - If commande 1
  - Then commande 2
  - Elif commande 3
  - Then commande 4
  - Elif commande 5
  - …
- Opérateur « et » :
  - commande1&&commande2
  - <==>if commande1 then commande 2 fi
- Opérateur « ou » :
  - cmd1||cmd2 : la commande 2 sera exécutée uniquement si la commande 1 se termine sur un code d'erreur ;

La programmation Shell (16)

- **Structure case :**
  - **Permet de sélectionner une action en fonction de certains mots.**
    - Case chaine in
      - Motif1) commande1;;
      - Motif2) commande2;;
      - …
      - Motifn) commanden;;
    - Esac
  - **Un motif peut être une expression régulière ;**
  - **Le symbole « | » permet de simuler le « ou » pour un motif ;**
  - **<u>Exercice</u> : écrivez un script qui traduit en anglais les chiffres de « zero » à « cinq ». Le script répondra « inconnu » en cas d'erreur.**

La programmation Shell (17)

- **While et until**
  - **While permet de répéter une commande tant qu'une condition est vraie ;**
    - While commande1
    - Do commande2
    - Done
  - **Until permet de répéter une commande tant qu'une condition est fausse**
    - Until commande1
    - Do commande2
    - Done
  - **<u>Exercice</u> : en utilisant la commande « who », écrire un script qui écrit une virgule à l'écran toutes les 10 secondes tant que l'utilisateur passé en paramètre est connecté.**

La programmation Shell (18)

- **Commande test :**
  - **Test évalue une expression et retourne vrai ou faux :**
    - Test expression
    - [expression]
  - **Utilisations :**
    - Test -w fichier : vrai si fichier existe et est autorisé en écriture ;
    - Test -r fichier : idem en lecture ;
    - Test -x fichier : idem en exécution ;
    - Test -d fichier : vrai si fichier existe et est un répertoire ;
    - Test -f fichier : vrai si fichier existe et n'est pas un répertoire ;
    - Test -s fichier : vrai si fichier existe et n'a pas une taille nulle ;

La programmation Shell (19)

- **Utilisation sur les chaînes :**
  - Test -z s1 : vrai si la chaîne s1 est vide (longueur de 0 caractères) ;
  - Test -n s1 : test inverse du précédent ;
  - Test s1 = s2 : vrai si s1 et s2 sont identiques ;
  - Test s1 != s2 : vrai si s1 et s2 sont différentes ;
- **Utilisation sur les nombres :**
  - Test n1 -eq n2 : vrai si n1 = n2 ;
  - Test n1 -ne n2 : test inverse ;
  - Test n1 -gt n2 : vrai si n1 est plus grand que n2 ;
  - Test n1 -lt n2 : vrai si n1 est plus petit que n2 ;
  - Test n1 -ge n2 : vrai si n1 est supérieur ou égal à n2 ;
  - Test n1 -le n2 : vrai si n1 est inférieur ou égale à n2 ;

La programmation Shell (20)

- **Combinaison de primitives :**
  - **Il est possible de combiner les primitives précédentes avec les opérateurs suivants :**
    - ! : négation ;
    - -a : ET logique ;
    - -o : OU logique ;
    - (*expression*) pour regrouper plusieurs expressions ;
  - **La priorité de -a est supérieure à celle de -o ;**
  - **Chaque primitive et opérande doit être séparée par un blanc ;**
  - **Les parenthèses doivent être protégées (« \ » ou « ` ») pour éviter d'être interprétées par le Shell ;**
  - **<u>Exercice</u> : écrivez un script qui déplace l'utilisateur dans le répertoire passé en paramètre et écrit un fichier vide dedans uniquement si ce répertoire existe. Un message d'erreur devra être affiché le cas échéant.**

La programmation Shell (21)

- **La commande expr :**
  - **Expr évalue une expression et retourne le résultat ;**
  - **Similaire à test, mais peut effectuer des calculs numériques :**
    - Expr e1 + e2 ;
    - Expr e1 - e2 ;
    - Expr e1 * e2 ;
    - Expr e1 / e2 ;
    - Expr e1 % e2 ;
  - **Faites le test suivant :**
    - a = 3
    - expr $a + 5

La programmation Shell (21)

- **Fonctions**
  - **Le shell permet la déclaration de fonction qui pourront être appelées au sein d'un script ;**
  - **L'appel d'une fonction est plus rapide que l'appel d'un script ;**
  - **Format :**
    - Fonction ()
    - {
    - Commande1
    - Commande2
    - }
  - **L'instruction « return n » permet de quitter la fonction en spécifiant le statut n.**

Travail Pratique

- **Exercice 9 : les scripts**
  - **Ecrivez un script qui affiche un menu donnant le choix entre 3 commandes :**
    - Affichage de la date ;
    - Addition de deux nombres ;
    - Quitter ;
  - **On se propose de créer une commande « trash » :**
    - Ecrivez un script « trash » qui déplace dans un répertoire « poubelle » les fichiers dont les noms sont passés en paramètres ;
    - Ajoutez une option -c à la commande permettant de connaître la taille de la poubelle (indice : utiliser la commande du) ;
    - Ajouter une option -e permettant de vider la poubelle ;
    - Ajouter une option -h permettant d'afficher l'utilisation de la commande ;

Travail Pratique

- **Exercice 9 : les scripts, suite.**
  - **Ecrire une commande shell avec 3 arguments. Elle indiquera si les 3 chaînes sont identiques ou non. par le code de retour suivant :**
    - 0 si les 3 chaînes sont égales ;
    - 1, 2 ou 3 si la chaîne différentes des autres est à la (1/2/3)ieme position ;
    - 4 si les trois chaînes sont différentes ;
    - 5 si le nombre de paramètres est incorrect ;
  - **Ecrire un script shell qui affiche le nom de tous les fichiers du répertoire /usr/include dont le nom se termine par ".h" et ayant plus de 100 lignes.**
  - **Ecrire un programme permettant de changer facilement l'extension d'une série de fichiers.**
    - Par exemple, "rename htm html ~/mydir" renomme tous les fichiers ".htm" du répertoire "mydir" en ".html".

Plan

- Généralités
- Premier contact avec UNIX
- Le système de Fichiers
- Les expressions régulières et les filtres
- Les processus
- Les scripts
- Sed et Awk

Sed

- **Sed : Stream Editor**
  - **Sed est un éditeur de texte non interactif ;**
  - **Permet d'appliquer des commandes à un fichier texte et d'afficher le résultat sur la sortie standard ;**
  - **Commandes :**
    - s : substitution. La plus utilisée ;
    - d : supression ;
    - a, i : insertion ;
    - q : quitte ;
    - = : écrit les numéros de ligne ;
    - w : écrit dans un fichier ;
    - ! : négation ;

Sed(2)

- **Utilisation de Sed :**
  - **Format général : sed commande fichier**
  - **Pour obtenir un fichier de sortie : rediriger la sortie standard ;**
  - **Commande s :**
    - Utilisation classique : sed s/ancien/nouveau/option fichier
    - Il est conseillé d'entourer la commande avec des apostrophes pour éviter que le Shell interprète les caractères spéciaux ;
    - Exemple :
      - sed 's/DiLib/DILIB/' log.txt > log_new.txt
  - **<u>Exercice</u> : créez un fichier catalogue de votre répertoire racine en remplaçant tous les 'a' par des 'A' à l'aide d'une commande sed.**

Sed(3)

- **Utilisation de Sed**
  - **Il est possible d'utiliser les expressions régulières dans la spécification des chaînes à remplacer ;**
    - Exemple : que fait cette instruction :
      - ls | sed 's/[abc]/'?'/g'
  - **Pour faire deux remplacements sur la même ligne de commande : utiliser -e**
    - Exemple : sed -e 's/*txt/BOUM/g' -e s/*log/CRASH/g' old.txt
  - **Exercice :**
    - Dans votre répertoire d'accueil, remplacez tous les débuts de ligne par « M » ;
    - Dans votre répertoire d'accueil, remplacez tous les mots qui commencent par A ou a par "remplaceA" et tous ceux qui commencent pas B ou b par "remplaceB".

Sed(4)

- **Utilisation de Sed**
  - **Il est possible de mémoriser des expressions régulières complexes en les encadrant avec les opérateurs « \(» et « \) ».**
  - **Les expressions mémorisées sont ensuite rappelées par \1, \2…**
  - **Exercice :**
    - Créez un fichier avec une liste d'étudiants sous la forme « Nom, Prénom ». On place un nom par ligne ;
    - Traitez ce fichier avec Sed pour qu'il l'affiche sous la forme « Prénom, Nom » ;

Sed(5)

- **Utilisation de Sed**
  - **Il est possible de rassembler les commandes sed dans un fichier ;**
    - # Fichier exemple.sed
    - # Commentaires uniquement au début
    - s/é/eaccentaigu/g
    - s/è/eaccentgrave/g
    - s/à/aaccentgrave/g
  - **Pour utiliser ce fichier, il faut utiliser l'option -f**
    - Sed -f exemple.sed fichier
  - **<u>Exercice</u> : créez un fichier qui transforme toutes les lettres minuscules en leur suivant lexicographique (cas particulier : z devient « ! »).**

Sed(6)

- **Utilisation de sed :**
  - **« & » : duplication de la chaîne trouvée**
  - **Exemple : que fait l'exemple suivant ?**
    - Echo 123 abc | sed 's/[0-9]*/&&'
  - **<u>Exercice</u> : Dans votre répertoire racine, encadrez tous les mots commençant par m avec des parenthèses ;**
  - **Option n et p :**
    - L'option -n, combinée avec p, n'imprime que les lignes modifiées.
    - Exemple : sed -n 's/pat1/pat2/p' fichier
  - **<u>Exercice</u> : reproduisez la commande grep à l'aide de Sed.**

Sed(7)

- **Utilisation de Sed :**
  - **Il est possible de restreindre les lignes à traiter :**
    - Pour une ligne : le numéro de ligne avant la commande :
    - Pour un intervalle : idem, sous la forme ligne_début,ligne_fin
      - Exemple : remplacez dans votre répertoire d'accueil, a par A mais uniquement sur les 10 premières lignes.
    - Pour remplacer la dernière ligne : « $ »
    - L'intervalle peut être constitué par des mots clés :
      - Exemple : sed '/start/,/stop/ s/#.*//
        - Que fait cette commande ?
    - Exercice : Ecrire une commande qui enlève tous les commentaires dans un fichier sed sauf entre les deux mots clés « start » et « stop ».

Sed(8)

- **Utilisations de Sed :**
  - **La négation : « ! ».**
    - Dans ce cas, la commande s'applique à toutes les lignes qui ne correspondent pas à la caractérisation ;
    - Exemple : sed -n 11,$!p affiche les lignes 1 à 10 ;
  - **La suppression : d**
    - Efface les lignes
    - Exemple : sed 1,10 d fichier : affiche fichier à partir de la 11eme ligne ;
  - **L'insertion : a et i**
    - a\ texte écrit le texte après la ligne ;
    - i\ texte écrit le texte avant la ligne ;

Awk

- Awk permet, tout comme Sed, d'appliquer un certain nombre d'actions sur un fichier.
- Sa syntaxe est inspirée du C ;
- Syntaxe :
  - awk [-Fs] [-v variable] [-f fichier de commande] 'programme' fichier
  - -F spécifie les séparateurs de champs ;
  - -v définit une variable utilisée à l'intérieur du programme ;
  - -f les commandes sont lues à partir d'un fichier ;