ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

Systèmes Informatiques
Interpréteurs de commandes
&
shells

G.BERTHELOT

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

Objectifs des commandes

- manipuler des fichiers
- modifier l'environnement
- lancer des applications (créer des processus)

syntaxe des commandes

- interpréteur bash (existent aussi sh, tcsh,..)
  - action [paramètre ou option]*

    (le nombre de paramètres ou options est variable, même pour une commande donnée, l'ordre des paramètres et option peut avoir de l'importance)
    - paramètre : objet sur lequel agir
    - option : précision sur l'action

Ensemble de fichiers

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

- l'ensemble des fichiers est structuré en une "arborescence"qui posséde
  - une racine unique
  - des feuilles qui sont des fichiers
  - des nœuds intermédiaires nommés répertoires (directories) contenant des répertoires ou des fichiers

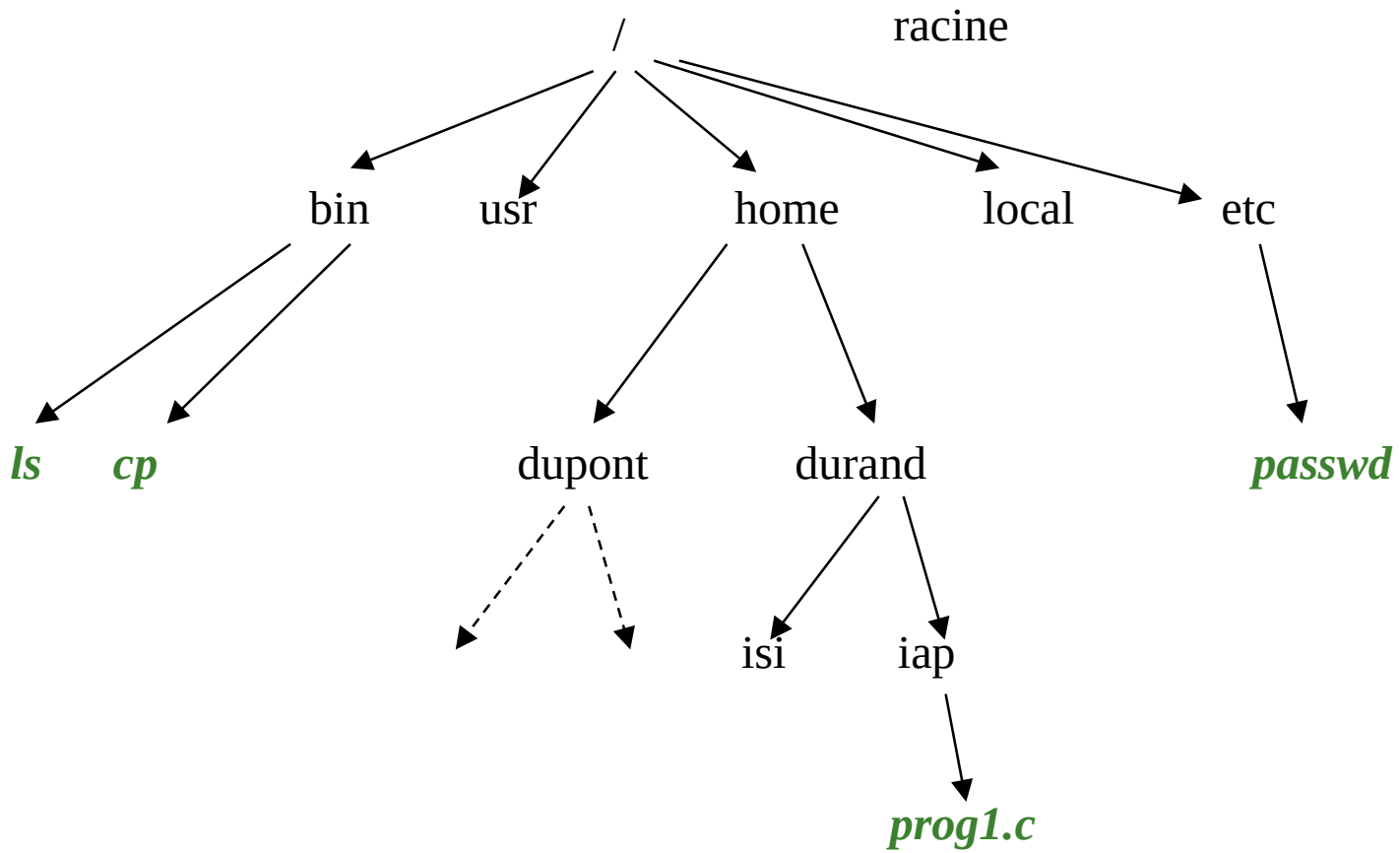
Ensemble de fichiers

- Trois types de fichiers
  - fichiers réguliers *tableaux d'octets*
  - répertoires
  - fichiers spéciaux *représentent les périphériques*

    exemples : /dev/hda, /dev/fd0, /dev/usb0

  un nom de fichier désigne un inode qui est un descripteur de fichier (propriétaire, droits d'accès, taille, date de création, de modifications, etc ...)

Arborescence des fichiers



Arborescence des fichiers

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

- Désignation d'un fichier (nom absolu):
  - succession des répertoires depuis la racine, puis nom (local) du fichier dans le dernier répertoire

  exemple: /home/durand/iap/prog1.c
  - avantage : possibilité d'avoir des noms locaux identiques dans des répertoires différents mais désignation unique
  - inconvénient : lourdeur (longueur des noms)

Ensemble de fichiers

- Faciliter la désignation d'un fichier : nom relatif :
  - répertoire de travail (working directory) ou répertoire courant (current directory)
    - l'utilisateur peut définir un répertoire de travail. Si un nom de fichier ne commence pas par / il est recherché dans le répertoire de travail
  - répertoires de recherche des actions (PATH)
    - Les fichiers correspondants aux actions demandées dans les commandes sont recherchés dans les répertoires donnés dans le PATH

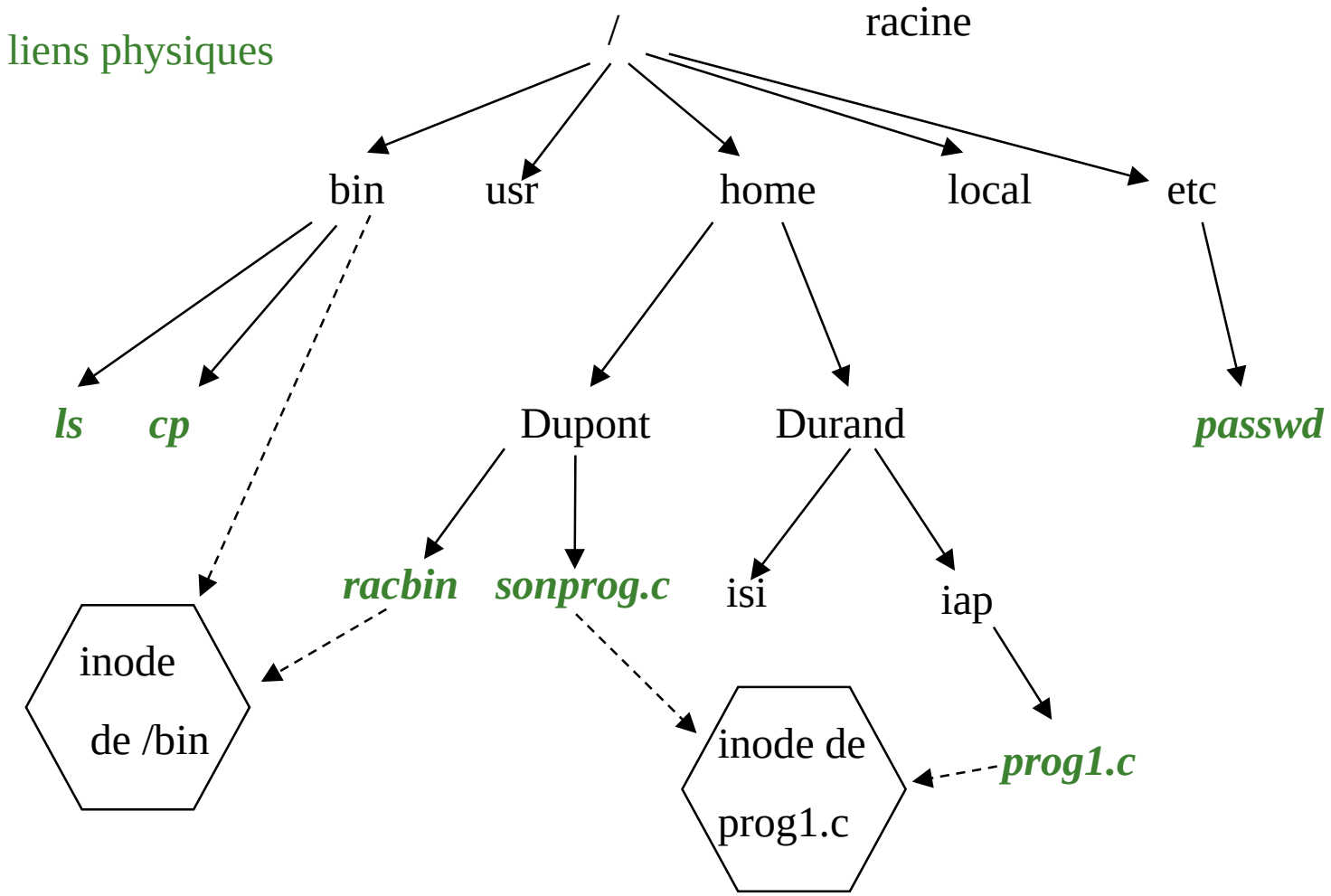
Ensemble de fichiers

- Faciliter la désignation des fichiers :
  - lien physique
    - l'utilisateur peut donner un second (troisième,…) nom à un fichier existant

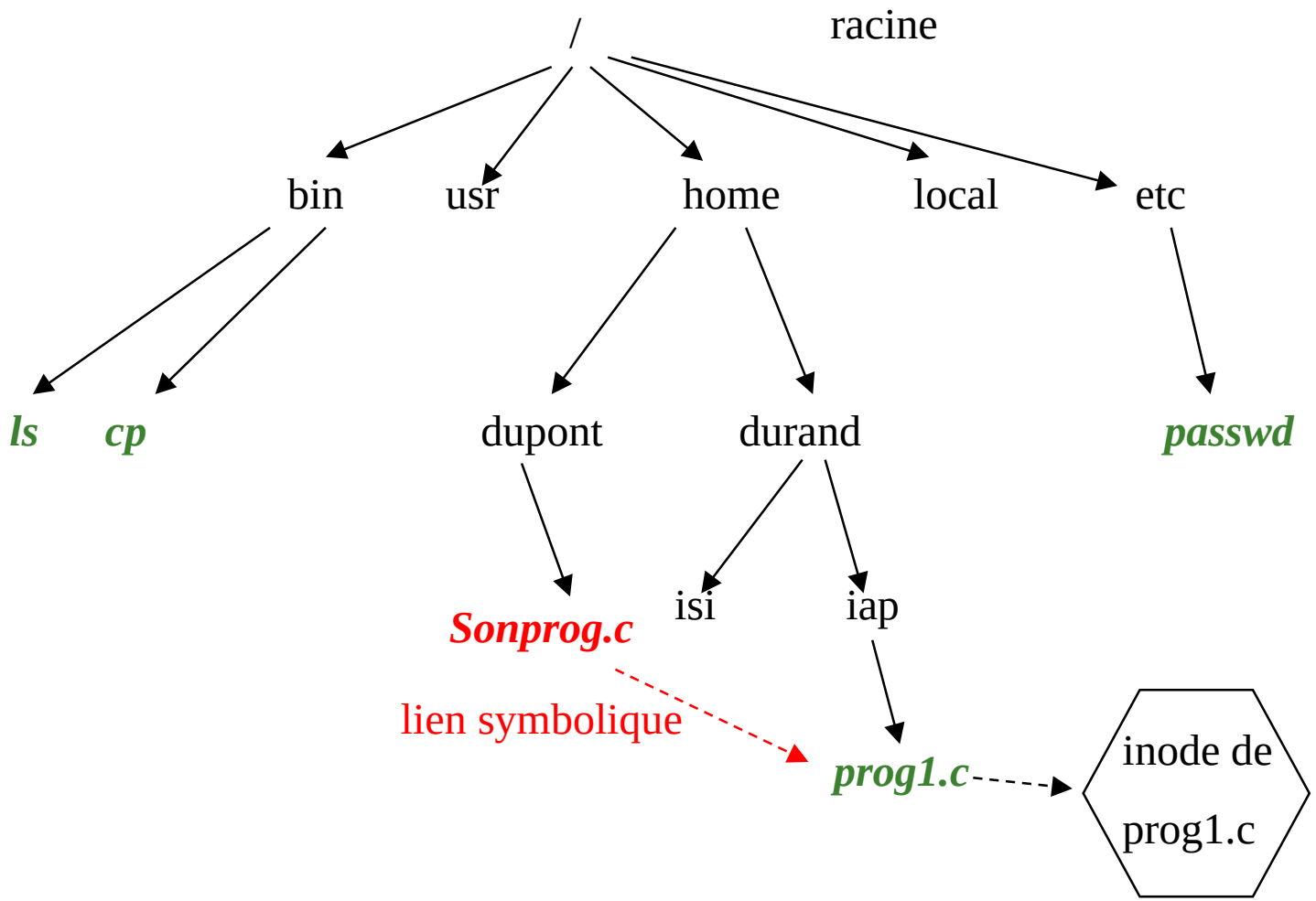
    ln /home/durand/iap/prog1.c /home/dupont/sonprog.c
    - limitation pour les liens sur répertoire
  - lien symbolique
    - l'utilisateur peut faire pointer un nom de fichier sur un autre nom de fichier

    ln -s /home/durand/isi/prog1.c /home/dupont/sonprog.c

Arborescence des fichiers



Arborescence des fichiers



Manipulation de fichiers

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

- cp <fichier_source> <fichier_destination>
- touch <fichier> *création d'un fichier vide*
- rm <fichier> *suppression d'un fichier*
- mv <fichier_source> <fichier_destination>
  changement de nom
- find <lieu> <critère> <action>
  find / -name "lola *" -print
- chmod u+x script1
- chown berthelot.prof doc_lilo.ps

Gestion de l'arborescence

- mkdir rep *création du répertoire rep*
- rmdir rep *suppression du répertoire rep*
- cd rep *le nouveau répertoire de travail est rep*
  - remarques : cd - *revient au répertoire précédent*, cd

  revient au répetoire de base, ainsi que cd ~/
- ls *affiche la liste des fichiers présents dans le répertoire de travail mais n'affiche pas les fichiers de noms commencent par un "."*
- pwd *affiche le répertoire de travail*

commandes utiles

- clear *efface le contenu de l'écran*
- echo *affiche à l'écran son résultat*
- cat fichier *affiche le contenu du fichier*
- less fichier *affiche le contenu du fichier en pages successives*
- grep chaîne *recherche et affiche les lignes contenant une chaine de caractères donnés*
- alias *permet de définir une commande, le plus souvent un abrégé, mais aussi* alias cd rm *

Modification de l'environnement

- Les effets d'une commande dépendent de l'environnement d'exécution. Il est constitué d'un certain nombre de variables dont la valeur est toujours une chaîne de caractères
  - PWD *répertoire courant*
  - HOME *répertoire de login ou de base*
  - USER *nom de l'utilisateur*
  - HOSTNAME, SHELL, TTY
  - PATH
  - PS1, PS2 *prompts*
  - TERM
- la valeur d'une variable VAR est accessible par: $VAR

Consultation de l'environnement

- commande printenv ou env

exemple

- echo $<variable>
  - exemple

    echo $PWD *résultat identique à pwd*

initialisation de l'environnement

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

- lors du login de l'utilisateur le système:
  - initialise le répertoire de travail à partir d'un champs du fichier /etc/passwd. Le plus souvent c'est son répertoire de base donc /home/dupont
  - lance un premier programme à partir du fichier /etc/passwd , le plus souvent /bin/bash
- fichier d'initialisation dans le répertoire de base
  - .profile
  - .bashrc

modification de l'environnement

- PATH=$PATH:./ pas d'espaces autour du "="

  modification temporaire, disparait avec le processus
- export PATH=$PATH:./

La variable PATH sera modifiée pour tout les processus créés à partir de ce moment, mais l'effet disparait au logout

- read VAR

  affecte à la variable VAR la suite des chaînes de caractères entrées sur l'entrée standard (le clavier) jusqu'à la frappe de la touche "entrée" (ou return)
- exemple
- une variable qui n'a pas été affectée vaut chaîne vide

exécution des commandes

- flux
  - flux sortants
    - la plupart des commandes produisent des suites d'octets.
      - exemple : ls produit une liste des noms des fichiers ou répertoires dans le répertoire courant.
    - on appelle une telle suite le flux sortie standard,
    - certaines commandes produisent un flux sortie erreur
    - sauf modification, le flux sortie standard et le flux sortie erreur sont envoyés sur l'écran de login

exécution des commandes

- flux
  - flux entrant
    - certaines commandes ont besoin d'un flux entrant.
      - exemple : read,
      - mais aussi cat, less,grep si aucun fichier n'est spécifié)
    - on appelle une telle suite un flux entrée,
    - Sauf mention contraire le flux entrée provient du clavier.

exécution des commandes

- flux
  - toutes les commandes ont a leur disposition:
    - un flux entrée standard (source : clavier),
    - le flux sortie standard (destination : écran)
    - le flux sortie erreur (destination écran),

    même si elles ne les utilisent pas
  - chacun de ces flux peut être rédirigé
    - flux entrée sur un fichier ou un périphérique
      - pour faciliter la saisie d'un flux important ou le répéter
    - flux sorties sur des fichiers ou des périphériques
      - pour éviter de submerger l'utilisateur et faciliter l'analyse

exécution des commandes

- redirections des flux
  - redirection du flux entrée : "<"
    - exemple: read < fich VAR

    (la source d'octets est fich et non le clavier)
  - redirection du flux sortie standard : ">"
    - exemple: ls >fich

    (résultat de ls envoyé dans fich et non à l'écran)
  - redir. sans écras. du flux sortie standard : ">>"
    - exemple: ls >>fich

    (résultat de ls est ajouté à la fin de fich)
  - redirection du flux sortie erreur : "2>"

exécution des commandes

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

- redirections des flux
  - Remarque

    redirection du flux d'entrée et redirection du flux de sortie peuvent être combinées

    exemple :

exécution des commandes

- commandes internes et commandes externes
  - **commandes internes** : le travail demandé est fait par l'interpréteur de commande lui-même
  - **commandes externes** (les plus nombreuses) l'interpréteur de commande ne fait pas lui-même le travail demandé. Il lance l'exécution d'un programme existant puis attend que ce dernier se termine
    - exemple : la commande ls provoque l'exécution du fichier exécutable /bin/ls

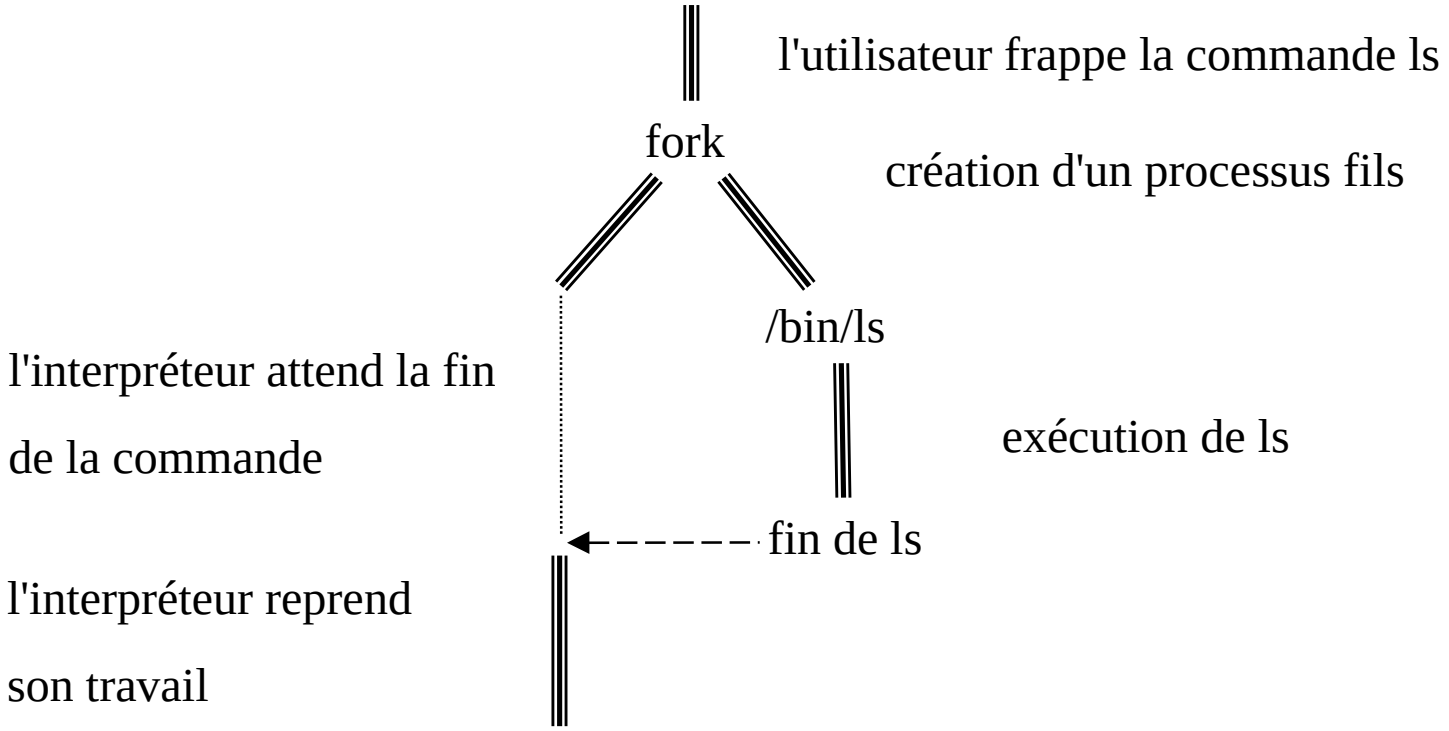
exécution des commandes

- intérêt des commandes externes
  - faciliter le développement d'un interpréteur
  - récupération des commandes existantes
  - extensibilité de l'interpréteur
    - on peut ajouter n'importe quel programme mis au point par un utilisateur
  - indépendance de l'interpréteur par rapport à la commande

    si la commande échoue l'intrepréteur reste intact et peut reprendre le dialogue avec l'utilisateur

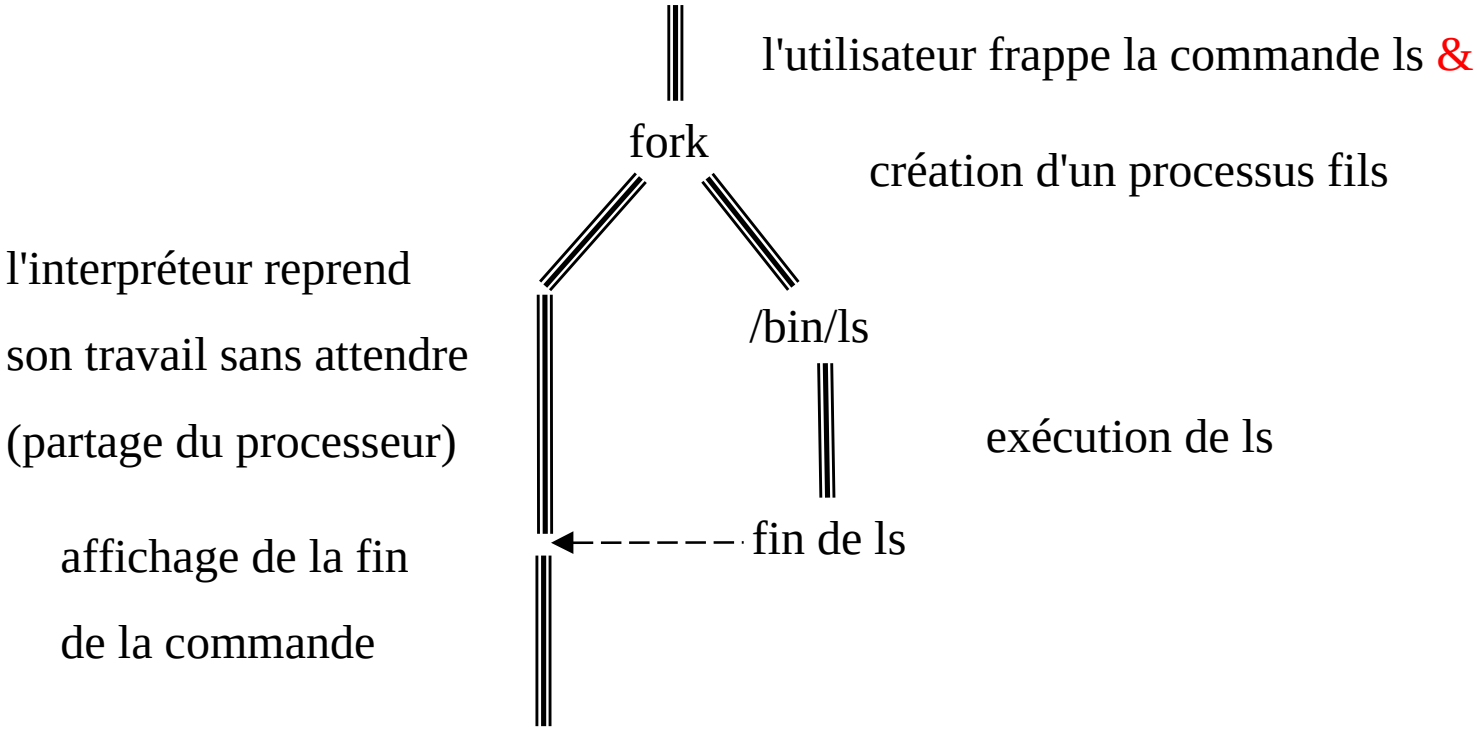
exécution des commandes

- transmission du contrôle d'exécution pour les commandes externes



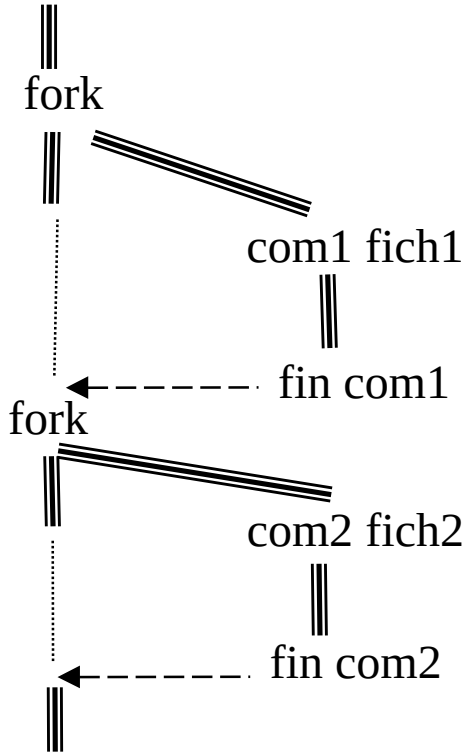
exécution des commandes

- transmission du contrôle :
  - tâche de fond ou background: symbole "&"



exécution des commandes

- enchaînement de commandes
  - en séquence : symbole ";"
    - exemple: com1 fich ; com2 fich

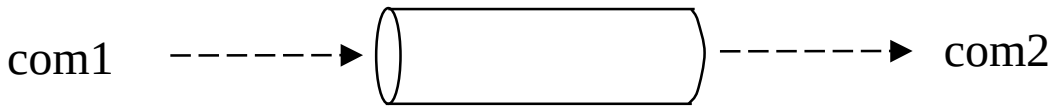


exécution des commandes

- enchaînement de commandes

  - enchaînement en coopération : symbole " | "
    - exemple : com1 [fich] | com2 [fich]

  (création d'un fichier intermédiaire ou tampon)



fichier intermédiaire: tube (pipe)

FIFO d'octets de capacité limitée

com2 est bloquée en attendant les octets produits par com1

exécution des commandes

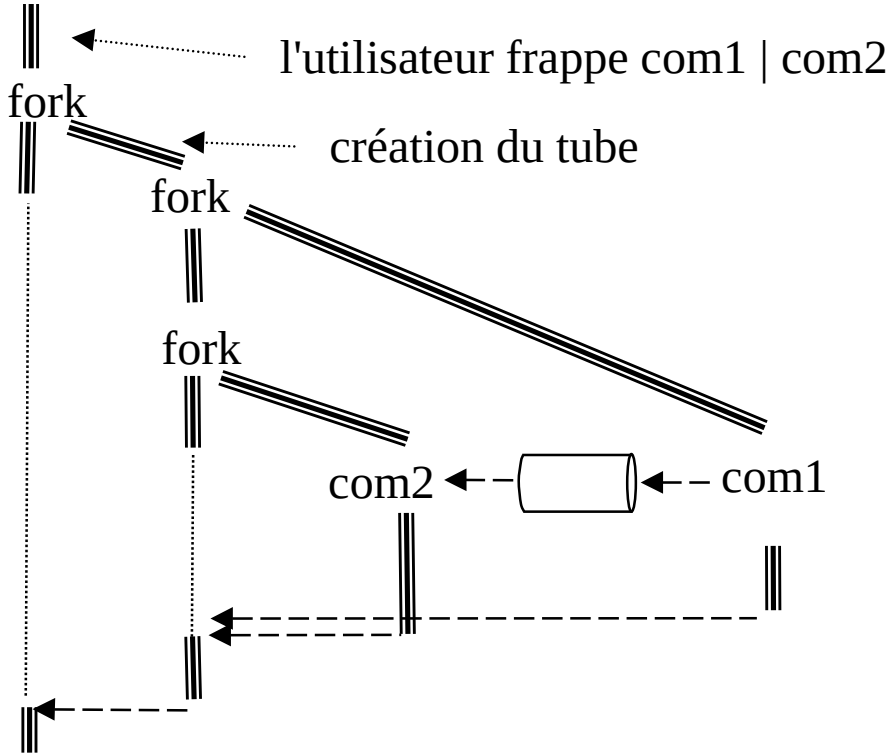
- enchaînement de commandes

  - enchaînement en coopération : symbole " | "
    - exemple : recherche du numéro de processus qui exécute le programme monprog

    ps -x | grep monprog

exécution des commandes

- enchaînement de commandes
  - passage du contrôle en coopération



exécution des commandes

- avant d'exécuter une commande l'interpréteur procéde à des modification de celle-ci. deux types de modifications :
  - remplacement des chaînes de caractères par leur valeur
  - si demandé exécution de la chaîne, puis remplacement de celle ci par son flux sortie

exécution des commandes

- remplacement des chaînes de caractères dans les commandes (désignation approchée)
  - dans un paramètre le caractère * vaut pour toute suite de caractères sauf le point (.) initial

    > ls

    >prog1.c prog2.c prog10.c pro12.c fich1.c

    >rm pro*.c

    est remplacé par :

    rm prog1.c prog2.c prog10.c pro12.c
  - quels seront les effets de rm * ? rm *.* ?

exécution des commandes

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

- remplacement des chaînes de caractères dans les commandes
  - dans un paramètre le caractère ? vaut pour tout caractère sauf le point (.) initial

    >ls

    >prog1.c prog2.c prog10.c pro12.c fich1.c

    >rm prog?.c

    est remplacé par :

    rm prog1.c prog2.c

exécution des commandes

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

- remplacement des chaînes de caractères dans les commandes
  - dans une paramètre l'ensemble [aox] vaut pour un des caractères qu'il contient

    >ls

    >pra.c prb.c prc.c prx.c fich1.c

    >rm pr[aox].c

    est remplacé par

    rm pra.c prx.c

  l'ensemble [a-z] permet la substitution de tout l'alphabet

exécution des commandes

- remplacement des chaînes de caractères dans les commandes

  lorsqu'une chaîne est précédée de $ l'interpréteur considère que c'est un nom de variable et la remplace par sa valeur :

  >exports APP_DIR=/home/berthelot/bin

  >cd $APP_DIR

  sera transformé en

  >cd /home/berthelot/bin

exécution des commandes

- délimitation et évaluation des chaînes de caractères dans les commandes
  - le caractère espace est considéré comme un délimiteur par l'interpréteur de commande
    - difficile de mettre un espace dans un nom de fichier

    exemple

    >cp monfichier mon fichier *echec!*
    - il faut utiliser les délimiteurs " "

    >cp monfichier "mon fichier" *succès!*

exécution des commandes

- délimitation et évaluation des chaînes de caractères dans les commandes
  - le caractère $ provoque le remplacement d'une variable par sa valeur
    - difficile de mettre un $ dans un nom de fichier

    exemple

    >cp "mon compte bancaire" mes$amoi *echec!*
    - il faut utiliser les délimiteurs ' '

    >cp "mon compte bancaire" 'mes$amoi' *succès!*
    - les délimiteurs ' ' empêchent les remplacements

exécution des commandes

- évaluation des chaînes de caractères dans les commandes : apostrophes inversées ` ` (backquote)
  - règle de base :
    - la première chaîne de la ligne de commande est considérée comme une commande et exécutée;
    - les chaînes suivantes peuvent être remplacées mais ne sont pas exécutées (sauf dans les structures de contrôle, voir plus loin)
  - les délimiteurs ` ` et eval permettent l'éxecution
    - export contenu_rep=`ls` ou encore
    - export contenu rep=eval ls

exécution des commandes

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

- Les caractères * ? $ [ ] jouent chacun un rôle spécial

- Si l'on désire les utiliser en tant que caractères normaux, il faut les faire précéder d'un caractère \ (backslash) qui devient ainsi spécial lui-même mais perd sa particularité lorsqu'il est doublé.

  mécanisme d'échappement

exécution des commandes

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

- valeur de retour d'une commande
  - une commande peut retourner une valeur (numérique)

    exemple : la commande

    exit 56

    retourne la valeur 56
  - la valeur de retour 0 est considérée comme vrai par l'interpréteur
  - Les valeurs de retour 1 et supérieures sont considérées comme faux par l'interpréteur

commandes sur les processus

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

- qu'est-ce qu'un processus ou tâche?

  un processus correspond à l'exécution d'une tâche bien définie : typiquement exécution d'une commande simple (exemple recopie d'un fichier, édition d'un fichier),
- différence entre programme et processus : un programme peut être lancé plusieurs fois en même temps -> autant de processus.
- le processeur et les ressources matérielles ou logicielles sont allouées à des processus.

commandes sur les processus

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

- ps

  affiche la liste des processus de l'utilisateur qui a fait le login, nombreuses options, -x est la plus intéressante

- fg numéro_processus

  ramène en foreground un processus qui s'éxecutait en background

- kill numéro_signal, numéro_processus

  envoie le signal indiqué au processus indiqué

- pkill nom *tue les processus dont le nom est donné*

Scripts

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

- pour éviter de répéter des suites fastidieuses de commandes on peut placer les commandes qui la composent dans un fichier,
- puis demander à l'interpréteur de commandes d'exécuter le fichier, c'est à dire d'exécuter la succession de commande.
- un tel fichier prend alors le nom de script, ou script shell ou même shell
- pas d'extension particulière (souvent .sh)

Scripts:exemple introductif

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

```
#!/bin/sh          ligne facultative

set `ls`
for i in $*
do
    if [ -d $i ]
    then echo "$i est un repertoire"
    fi
    if [ $i = "fi_lo" ]
    then echo "fi_lo trouve. Voir son contenu ?"
         read rep
         case $rep in
             o | O ) cat $i;;
             n | N ) echo "pas de visualisation de fi_lo";;
             * ) echo "vous repondez n importe quoi"
         esac
    fi
done
```

Scripts

- créer et exécuter un script monscript
  - créer et remplir un fichier monscript avec un éditeur de texte
  - le rendre exécutable avec chmod u+x monscript
  - l'exécuter en soumettant son nom à l'interpréteur de commandes
    - exemple

    >monscript fich1 fich2
  - ou aussi

    >bash monscript fich1 fich2

Scripts

- principaux éléments du langage de script
  - paramètres positionnels
  - variables : les variables d'environnement, + des variables locales si besoin est, + paramètres
  - instructions : les commandes reconnues par l'interpréteur de commandes
  - structures de contrôle usuelles
    - boucles pour itération ou répétition
    - expression booléennes
    - if then else fi et case (aiguillage ou switch)

Scripts

- Un *script* peut avoir des paramètres dits **positionnels**, qui sont alors passés dans la ligne de commande:
- **monscript param_1 ... param_n**.
- Dans les commandes du fichier **monscript** **$1** désigne la valeur du premier paramètre
- **$2** désigne le 2ème, ..., **$9** le 9ème

Scripts

- la commande **shift** supprime les 9 premiers et décale les autres
  - shift

  $1 prend la valeur de $10, $2 celle de $11, etc...
- la commande **set** réaffecte les paramètres positionnels

  set ch_1 ch_2 ch_3

  $1 vaut ch_1, $2 vaut ch_2 , $3 vaut ch_3 et $4 a pour valeur la chaine vide

Scripts

- paramètre positionnels ajoutés :
  - **$0** : *nom de la commande appelée*

    (nom du fichier contenant le script)
  - **$*** : *liste des paramètres de la commande* ;
  - **$#** : *nombre de paramètres de la commande*;
  - **$$** : *numéro du processus qui exécute la commande ;*
  - **$?** : *code de retour de la dernière commande exécutée.*

Scripts

- Expressions booléennes : commande test
  - test arg1 arg2 ...

renvoie une valeur interprétée par vrai ou faux, et peut donc être utilisée dans les instructions conditionnelles

test $var = OUI

teste si la valeur de var *est* OUI

Scripts

- Expression booléennes: opérateurs de test
  - un des paramètres de la commande test est un opérateur. Il est binaire (infixe) ou unaire (préfixe)
  - opérateurs binaires pour comparer des chaînes :
    - = , !=
  - opérateurs unaires sur les chaînes de caractères
    - -z *(pour test chaîne vide)*, -n *(pour test chaîne non vide)*
  - opérateurs binaires pour comparer des nombres:
    - -eq *(equal)*, -ne *(not equal)*, -gt *(greater)*, -lt *(less than)*, -ge *(greater or equal)*, -le *(less or equal)*

Scripts

- Expression booléennes : opérateurs de test
- Tests sur les fichiers :
  - **-e fich** *fich existe-t-il comme fichier régulier ou fichier spécial ou répertoire*
  - **-d fich** *fich est-il un répertoire*,
  - **-f fich** *fich est-il un fichier régulier*
  - **-r fich** *la commande a-t-elle permis. de lire fich*,
  - **-w fich** *la commande a-t-elle permis. d'écrire fich*,
  - **-x fich** *la commande a-t-elle permis. d'éxecuter fich*.

Scripts

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

Structure de contrôle : instruction conditionnelle

```
if  test $var = OUI
then cp fich1 fich2
else echo rien fait
fi
```

- les termes if, then, else, fi doivent être sur des lignes différentes ou séparés par des ; sinon erreur syntaxe

```
if test $var = OUI ; then cp fich1 fich2
else echo rien fait
fi
```

Scripts

Structure de contrôle : instruction conditionnelle

forme générique :

if <liste de commandes_1>

then <liste de commandes_2>

else <liste de commandes_3>

fi

- <liste de commande> :: com_1; com_2;..; com_n
- la valeur retournée par une liste de commande, notament pour la condition, est la valeur retournée par la dernière commande de la liste (ici com_n)

Scripts

Structure de contrôle : instruction conditionnelle

Variante : utiliser une expression entre [ ]

```
if [ $var -eq OUI ]        #attention : mettre des espaces
then cp fich1 fich1
else echo rien fait
fi
```

Scripts

Structure de contrôle :

Remarques

Les commandes des listes ne doivent pas être mise entre backquotes, l'éxecution est automatique!

Le résultat de l'exécution de la chaîne true renvoie true et l'exécution de la chaîne false renvoie false

Pour la valeur d'une variable

- toute chaîne de caractères non vide vaut true,
- seule la chaîne vide vaut false

Scripts

Structure de contrôle : aiguillage

case <chaîne de caractères> **in**

<motif1> **)** <liste de commandes 1> **;;**

...

<motif n> **)** < liste de commandes n>**;;**

***)** <liste de commandes pour les autres cas> **;;**

esac

- remarques
  - un motif est une chaine de caractères;
  - * est égale à toute chaîne de caractères
  - liste_commande utilise des ; -> ne pas oublier les ;;

Scripts

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

Structure de contrôle : répétition

for <variable> **in** <liste_chaînes_caractères>

do <liste de commandes>

done

Parcours de la liste des paramètres :

for <variable> **in** $*

do <liste de commandes>

done

Scripts

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

Structure de contrôle : répétition

while <liste de commandes 1>

do <liste de commandes 2>

done

until <liste de commandes 1>

do <liste de commandes 2>

done

Scripts

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

- Expressions arithmétiques
  - Les variables ayant une valeur arithmétique peuvent être utilisées dans des calculs
  - exemple
    - i=1 #attention : ne pas mettre d'espaces autour du =
    - i=$(($i+3))
    - i=$[$i+3]

Scripts

- Manipulations sur les chaînes de caractères
  - Les manipulations complexes (substitutions paramètrées) sur des chaînes sont faites le plus souvent avec des commandes spécialisées en utilisant des expressions rationnelles:
    - sed,
    - awk,
    - ...
- les extractions simples peuvent être facilement avec la commande cut

Scripts

- cut
  - cut extraie une partie de chaque ligne de son entrée (fichier mentionné, ou entrée standard) et l'écrit sur sa sortie standard.
  - Les opt. -b, -f ou -d spécifient le type d'extraction.

    -b liste_d_octets ou --bytes liste_d_octets

    extraie les octets aux rangs indiquées dans la liste_d_octets.

    exemple :

    ```
    hh=`echo "12:34:56"|cut -b4-5`
    ```

    affecte "34" à hh

Scripts

- cut
  - l'option -f (pour field) permet d'extraire des champs, un champs étant une suite de caractères encadrés par des délimiteurs. Les délimiteurs sont paramètrables par l'option -d

    -f liste_de_champs, --fields liste_de_champs

    -d, --delimiter séparateur

    exemple

    ss=`echo "12:34:56"|cut -d: -f3`

    affecte "56" à ss

Scripts

- conclusion
  - possibilité de faire des traitements récursifs
  - possibilité de définir des fonctions
- bibliographie
  - man bash