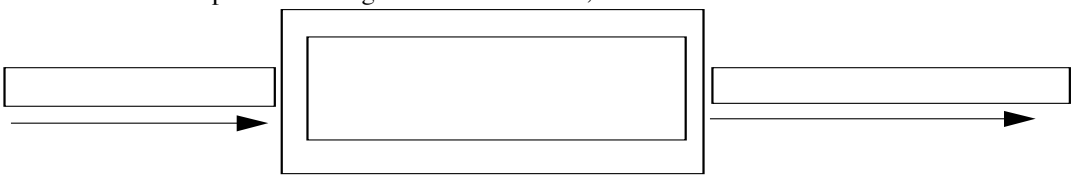
Cours 8
Utilitaires

Principe des flux de données

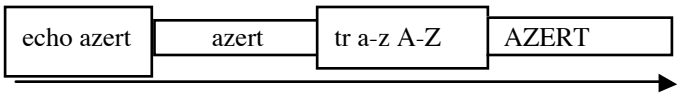
Notions fondamentales :

- flux de données :
  - chaque flux est connecté à un périphérique ou un fichier (en entrée ou en sortie). ex :

    ```
    FILE* stream1, stream2; // crée deux références de flux
    stream1  =  fopen("monfich.txt","r")  ; // connecte le flux 1 à un fichier
    stream2 = fopen("/dev/tty", "r") ;  // connecte le flux 2 au clavier
    ```
  - les données sont parcourues et générées dans l'ordre, sans retour arrière



- Pour mémoire : redirection vers des fichiers ou des périphériques : lors du lancement de la commande, on peut changer la connexion du flux :
  `ls –al >> listefich` (sortie standard redirigée vers un fichier, en ajout à la fin)
- tube : au lieu de rediriger la sortie standard vers un fichier ou un périphérique, on la connecte à un autre traitement dont elle devient l'entrée standard. Un même flux reçoit ainsi plusieurs traitements successifs. ex: echo azert | tr a-z A-Z



Exemples :

```
ls -al | more  # pour avoir le temps de lire
```

- substitution d'historique : la sortie standard de la partie entre quottes inverses (backquote, ` `) est redirigée vers le texte de la commande. Ex :

```
# renomme $nomfich en majuscules    ($nomfich est un paramètre)
mv $nomfich `echo $nomfich | tr a-z A-Z`
```

La conception Unix

1. Idée fondamentale : **ne pas avoir à réécrire un nouveau programme à chaque tâche ; on écrit des programmes qui font une seule chose** (afficher, ou formater, ou trier ou …), **mais de la façon la plus générale possible. On dispose ainsi d'une boite à outils d'utilitaires simples ; il y en a des centaines avec lesquels on peut faire presque toutes les tâches quotidiennes en les enchaînant par des tubes (pipe), des redirections, des substitutions de commandes.**

2) Quelques catégories d'utilitaires et quelques illustrations :

`alias` : permet de rebaptiser une commande complexe par un nom commode

Les utilitaires de renseignement sur les fichiers :
`test nomfich` : *est-ce que le fichier existe, a les droits de lecture, est un catalogue, etc. ?*
`file` : *quel est le type de contenu du fichier ?*
`wc` : *combien y a-t-il de caractères, de mots, de lignes dans le fichier ?*
`whereis, which` : *où sont les exécutables, lequel est lancé*
`basename, dirname` : *extrait le nom du fichier, son chemin d'accès*

Ex :

```
ls /dev | wc -l    # donne le nombre de drivers de périphérique
```

`set fullname=`which emacs` ;cd `dirname $fullname`` # se positionne dans le catalogue où est emacs
Les utilitaires de renseignement sur les utilisateurs
`who, who am I` : *qui travaille sur la machine en ce moment, sous quel nom suis-je logué*
`finger` : *renseignements sur un utilisateur (nom, nom de login, dernier accès, autres informations*

utilitaires d'affichage de texte :
`echo` : *affichage d'une chaîne de caractères*
`more, page, cat` : *affichage du contenu d'un fichier*
`od` : *affichage du contenu du fichier en octal, hexadécimal, binaire*
`head, tail` : *affichage des premières, des dernières lignes d'un fichier*
`cut, join` : *extraction de colonnes, fusion sur une colonne (opérations de BD)*
`grep` : *affichages des lignes d'un fichier qui contiennent (ne contiennent pas) un motif donné*
`tr` : *substitution de caractères*
`lp` : *envoi à une imprimante*

`alias usprint tail +10 /etc/passwd | cut –d: -f1,5 | lp` *# usprint* imprime la liste des utilisateurs non système
`alias whichmain 'grep –n main *.c'` *# whichmain* affiche les noms des fichiers C qui ont un main, avec la ligne où est le main et le numéro de cette ligne.

Quelques autres utilitaires

Utilitaires de comparaison et de tri de fichiers
diff (différences entre lignes) ; comm (lignes communes): cmp (caractères) ; uniq (supprime les lignes répétées) ; sort (tri de lignes) ; cut (extrait des colonnes : projection de bases de données) ; join (jointure de bases de données) ; paste (rassemble les lignes de même rang de chaque fichier)

Formateurs : expand, fold, fmt, cb, indent
Exemples :
`expand monfichier | fold -72 | page` # remplace les tabulations par des blancs et va à la ligne après 72 caractères
`fmt montexte | postprint –n2 –pland | lp -dlw5` # découpe le texte (ascii) en pages de 60 lignes, le traduit en postscript au format paysage 2 pages par feuille et envoie le résultat a l'imprimante lw5

recherche de fichiers : find

find parcourt récursivement une hièrarchie de fichiers. Pour chaque fichier recontré, find teste successivement les prédicats spécifiés par la liste d'option, jusqu'au premier qui échoue ou jusqu'à la fin de la liste. Principales options :
- name *nom*
-print (écrit le nom du fichier ; réussit toujours),
-exec (pour exécuter une commande. {} est le fichier courant. Terminer par \;),
-type (d: catalogue, f: fichier ordinaire, p: pipe, l: lien symbolique)
-newer *fichier* (compare les dates de modification)
-o (ou),
-prune (Si le fichier courant est un catalogue, élague l'arbre à ce point).

Exemples :
`find /usr/include -type d -print` # affiche les sous catalogues de /usr/include
`find /usr/lib -name '*.so*' -print` # liste les librairies dynamiques
`find . \( -type d -name source -prune \) -o -name '*.c' -exec mv {} ./source \;` #déplace tous les fichiers C dans le catalogue "source"

Le programme make

Syntaxe : `make [-f fichier] [but]`. "fichier " décrit un ensemble de buts, de dépendances et de commandes associées à ces buts. Si but n'est pas à jour, c'est à dire si ce n'est pas un fichier ou si c'est un fichier dont la date de dernière modification est antérieure à la date de dernière modification d'une de ses dépendances, make exécute les commandes associées à "but". La description des buts dans "fichier" a la forme suivante :

```
but : dépendance1 [...[dépendancen]]
      commande1
      commande2
      ...
```

Les dépendances peuvent être d'autres buts ou des fichiers. Par défaut, make utilise le fichier nommé `makefile` ou `Makefile` dans le catalogue courant et le premier but dans le fichier.
Exemples :
Le makefile du td 2 :

```
> cat makefile
mymain: mymain.o util.o        #  le but mymain dépend de mymain.o et de util.o
      gcc -o $@ $^             #  Si le but n'est pas à jour, on appelle gcc. $@ est le but,
                               #  $^  la liste des dépendances
```

On peut réutiliser ce fichier pour n'importe quel projet en changeant le nom du but et des dépendances.

Un système d'envoi de messages (dans un autre catalogue) :

```
> cat makefile
message :                      # but 1 : création d'un message
      provi=./tmp/mess$$$$     # création d'un fichier provisoire
      emacs $$provi            # édition du fichier pour écrire le message
      cat  ~/entete  $$provi  ~/signature > message   # ajout de l'en tête et de
                                                      # la signature
      rm $$provi               # suppression du fichier provisoire

modif :                                  # but 2 :modifier le message
      emacs message

envoi : message                # but 3 : envoyer le message
      mail eleves < message    # envoie à eleves un mail dont le contenu est message
      i=`ls envoi* | sed "s/envoi//" | sort -n| tail -1`   # cherche le
                                                  # numéro du dernier message archivé
      i=expr $$i + 1           # prochain numéro d'archive
      mv message envoi$$i      # renomme le message comme une archive
      cat /dev/null > message # crée un fichier 'message' vide
      rm message~              # supprime le fichier de sauvegarde
      touch envoi              # met à jour la date de modification de envoi
```

make message créée un nouveau message.
make modif édite le message, y compris l'en tête et la signature.
make envoi envoie le message mais seulement si il a été modifié depuis le dernier envoi.

Annexe 1 : quelques commandes de vi

Note : les italiques désignent un type de donnée ; *texte*, *nombre*, *char*,, *expreg* signifient donc qu'il faut entrer à cet endroit selon le cas du texte, un nombre, un caractère ou une expression régulière. CR signifie Carriage Return, ^N désigne la combinaison de touches (control + N), esc désigne la touche (escape).

commande	effet	quelques préfixes
a *texte* esc	insère texte après le curseur	*nombre* : répète texte
i *texte* esc	insère texte avant le curseur	*nombre* : répète texte
A *texte* esc	insère texte en fin de ligne	*nombre* : répète texte
i *texte* esc	insère texte avant le mot	*T5* : répète texte
o *texte* esc	crée une nouvelle ligne et insère texte	
↓, j ou ^n	ligne suivante, même colonne	*nombre* : répète le
↑,k ou ^p	ligne précédente, même colonne	déplacement
+, CR	premier mot de la ligne suivante	id
-	premier mot de la ligne précédente	id
^F, ^B	un écran en avant (en arrière)	
nombre z	fixe la taille de la fenètre	
H,M,L	En haut, au milieu, en bas de la fenètre	
nombre G	à la ligne *nombre*	
/*expreg*	à la prochaine ligne filtrée par *exprreg*	*nombre*…
?*expreg*	idem en recherche arrière	
n, N	répète la dernière recherche (même sens, sens contraire)	
0	au début de la ligne	
^	au début du premier mot de la ligne	
$	à la fin de la ligne	
h,^H,←	caractère précédent	*nombre* : répète le
l, space,→	caractère suivant	déplacement
f*char*	au prochain exemplaire de *char*	id
F*char*	idem en arrière	id
;	répète le dernier f ou F	id
,	id en inversanr le sens de recherche	id
nombre |	à la colonne *nombre*	
w, b, e	Début du mot suivant (précédent, fin du mot)	*nombre*, d, c, !
(,)	phrase précédente (suivante)	id
{, }	paragaphe précédent (suivant)	id
u	annule la dernière modification	
U	annule toutes les modifications de la ligne courante	
.	répète la dernière modification	
x, X	détruit un caractère sous (devant) le curseur	*nombre*
p,"*char*p	coller (depuis le tampon *char*)	
J	joindre des lignes	*nombre*
opérateurs (doublés ⇒ l'argument est la ligne)		
d, "*char*d	mettre à la corbeille (dans le tampon *char*)	
c	changer	
y,"*char*y	copier (dans le tampon *char*)	
!	filtrer par une commande externe	
:w	sauver	
:w *nom*	sauve sous le nom *nom*	
:wq	sauve et quite (synonyme ZZ)	
:q!	quitte sans sauver	
:e *nom*	edite le fichier *nom*	
e!	abandonne toutes les modifications depuis la dernière sauvegarde	