

# Le langage de commandes

# Le langage issu du traitement par lot

- Origine: cartes perforées

- Reconnaissance des cartes de commandes par caractères spéciaux: \$ ou // en début de carte

- Décomposition du *travail* (job) en *étapes* (step)

- Étape: programme à exécuter et liaisons avec objets externes

- 3 commandes

- identification du travail

- définition du programme et de son environnement

- définition d'une liaison fichier logique - objet externe

- Beaucoup de paramètres

- transmission par mot clé, valeurs par défaut

- macrocommandes (fichier) avec paramètres

# Le langage interactif

- Identification de l'utilisateur à la connexion (*login*)
  - valide pendant toute la session
  - terminée à la déconnexion (*logout*)
- Avec qui dialogue-t-on?
  - pas de commande pour Pierre, Paul ou Jacques, mais dire qui est à l'écoute
  - message d'invite (*prompt*): envoyé par le programme en attente
  - paramétrable par l'utilisateur
- Souplesse et convivialité
  - beaucoup de commandes, peu de paramètres
  - syntaxe orientée "verbe"
  - commandes reconnues par l'interpréteur exécutées directement
  - les autres => programmes quelconques de même nom = extensibilité

# Gestion de l'environnement

- Session de plusieurs heures
  - exécution de beaucoup de commandes
  - ne pas avoir à répéter les mêmes choses sans arrêt
- Environnement
  - informations gérées par l'interpréteur
  - utilisées pour modifier commandes et paramètres
  - transmises aux programmes
- Exemples
  - message d'invite est une variable d'environnement
  - répertoire de travail
  - règles de recherche (exécutables, manuels en ligne, bibliothèques...)
  - type du terminal et descriptions de ses possibilités

# Exemple d'environnement

`SHELL=/bin/csh` ← interpréteur de commande

`PATH=/usr/local/bin:/usr/bin/X11:/usr/bin:/usr/ccs/bin:/usr/ucb:/bin:/sbin:/usr/sbin:/usr/bin/mh` } règle de recherche d'exécutable

`LC_CTYPE=iso_8859_1` ← code des caractères

`TERM=xterm` ← terminal

`TERMCAP=vs|xterm|vs100:do=^J:le=^H:ho=\E[H:co#80:etc.` } capacités

`TZ=MET` ← fuseau horaire

`MANPATH=./usr/local/man:/usr/man` ← règle de recherche de manuel

`LD_LIBRARY_PATH=/usr/lib/X11` ← règle de recherche de bibliothèque

`MYSYS=alpha-OSF1` ← identification machine - système courant

# Traitement préliminaire

- Mécanisme de substitution

remplacement dans la commande des variables d'environnement

exemple: `$MYSYS/toto` devient `alpha-OSF1/toto`

interprétation des caractères spéciaux

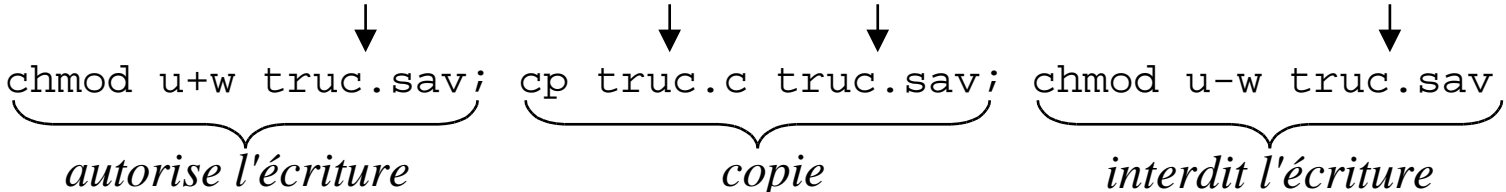
\* `.c` devient la liste des fichiers du répertoire courant se terminant par `".c"`

- Mécanisme d'abréviation

donner une forme plus simple à certaines commandes et fixer des paramètres par défaut

alias `rm 'rm -i'` => option par défaut `"-i"`

```
alias save 'chmod u+w \!:1.sav; cp \!:1.c \!:1.sav; chmod u-w \!:1.sav'
save truc
```



*autorise l'écriture*                      *copie*                      *interdit l'écriture*

# Exécution de la commande

- Recherche du programme de même nom (règles de recherche)

Chargement,

transmission des paramètres

transmission de l'environnement

lancement de son exécution

*éventuellement avec  
création d'un processus*

- liaisons avec les objets externes => paramètres ou environnement
- liaisons entrée standard, sortie standard, sortie d'erreurs
- redirection => permet de changer la liaison implicite terminal

*fichiers reliés par programme*

*fichier résultat*

*concatène les fichiers* → `cat *.biblio` | `sort` | `uniq` > `biblio.tri` → *supprime les doublons*

*tri l'entrée* → `sort`

# Les structures de contrôle (1)

- Appropriées aux objets manipulés: programmes et fichiers
- Commande conditionnelle:

```
if test -f toto
then echo "-----" >> toto
else echo "Etats des résultats successifs" > toto
fi
date >> toto
echo "-----" >> toto
```

*test existence du fichier "toto"*

*prolongement*

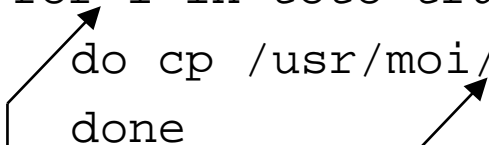
*création*



# Les structures de contrôle (2)

## ● Itération bornée

```
for i in toto truc bidule
do cp /usr/moi/$i /tous
done
```



*i vaut successivement*  
"toto", "truc", "bidule"

*copie des trois fichiers du*  
*répertoire " /usr/moi "*  
*dans le répertoire " /tous "*

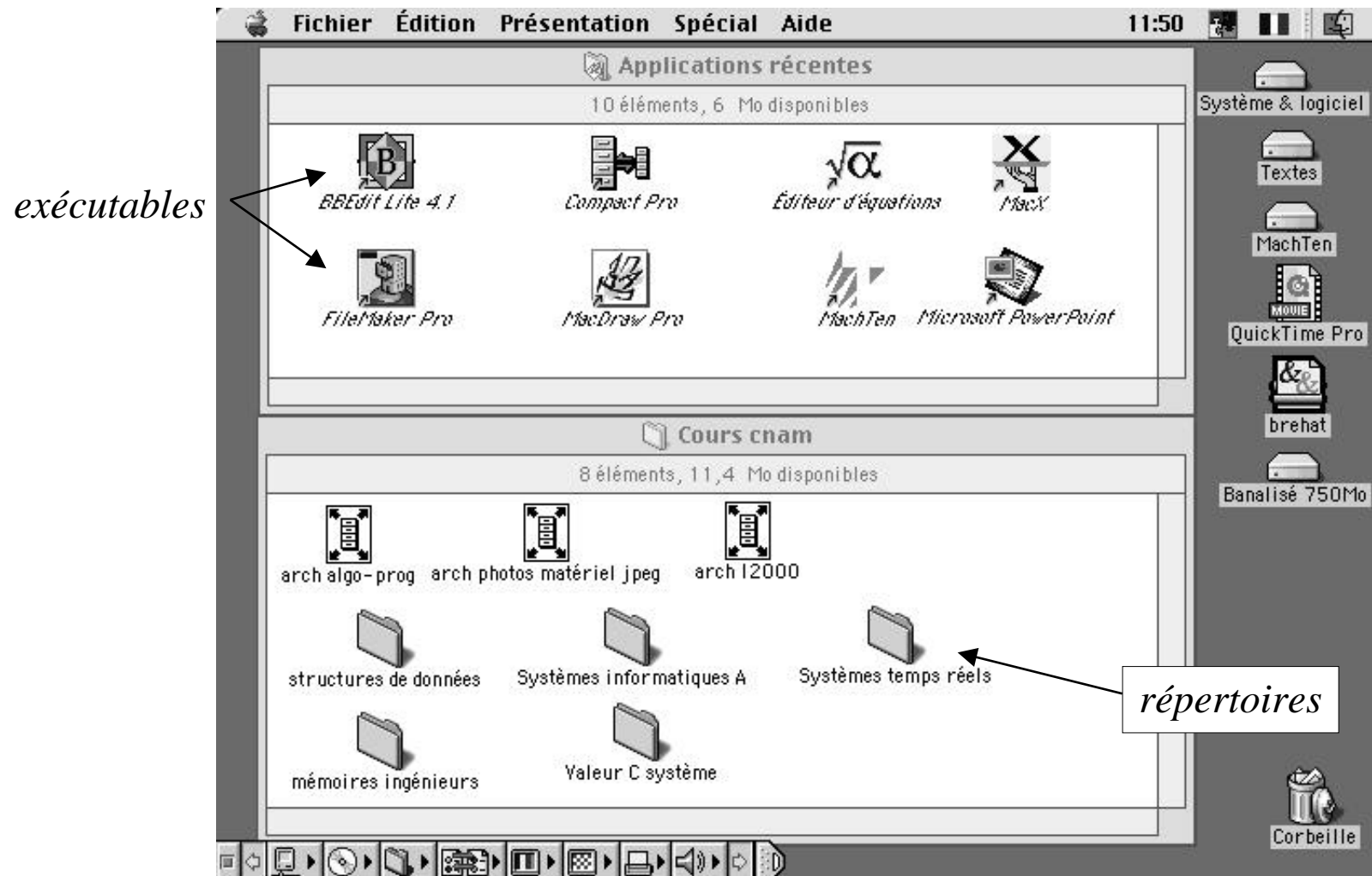
```
for i in *.c; do cc -c $i; done
```

*compilation de tous les*  
*fichiers en langage C du*  
*répertoire courant*

# Langage à base d'icônes

le langage de commandes

10



# Conclusion (1)

- Langage issu du traitement par lot
  - peu de commandes
  - beaucoup de paramètres
  - valeurs par défaut
- Langage interactif
  - convivial, orienté verbe
  - beaucoup de commandes, peu de paramètres
  - facilement extensible
- Environnement
  - informations conservées pendant la session
  - gérées par l'interpréteur, communiquées aux programmes

# Conclusion (2)

- Langage évolué
  - mécanismes de substitution, d'abréviation
  - redirection des entrées-sorties
  - structures de contrôles évoluées
- Menus déroulants et icônes
  - convivialité
  - ne peut satisfaire tous les besoins