

Outils système

présentation, shell & Emacs



Alexis Scheuer

Maître de conférences UL

FST, dépt info. & Loria



Plan

- Généralités
- Interpréteur de commandes
 - Fonctions
 - Spécificités de bash
- Emacs
 - Intérêt par rapport à d'autres éditeurs
 - Commandes simples
 - Aller plus loin





Format / séances

6 h de CM, 4 h de TD & 20 h de TP (cf DI)

16 – 20 janvier		CM 1, ve 20, 8 h
23 – 27 janvier	TP 1, lu 23 & ma 24	CM 2, ma 24, 16 h
30 janvier – 3 février	TP 2, lu 30 & ma 31	TP 3, me 1 ^{er} – ve 3
6 – 10 février	TP 4, lu 6 & ma 7	CM 3, lu 6, 10 h
13 – 17 février	TD 1, lu 13 & ma 14	TP 5, me 23
20 – 24 février	Vacances	
27 février – 3 mars	TD 2, lu 27 & ma 28	TP 6, me 1 ^{er}
6 – 10 mars	TP 7, lu 6 & ma 7	TP 8, me 8
13 – 17 mars	TP 9, lu 13 & ma 14	TP 10, me 15





Contenu

- Utilisation d'un système d'exploitation (Linux) et programmation en langage de script (bash, csh)
- Variables d'environnement
- Arborescence des fichiers, droits des fichiers
- Commandes simples de Unix/Linux, pipe et redirections des entres-sorties
- Notion de processus
- Expressions régulières et outils Linux de filtrage
- Outils de développement d'applications (configuration, gestion de versions, ...)





Commandes considérées

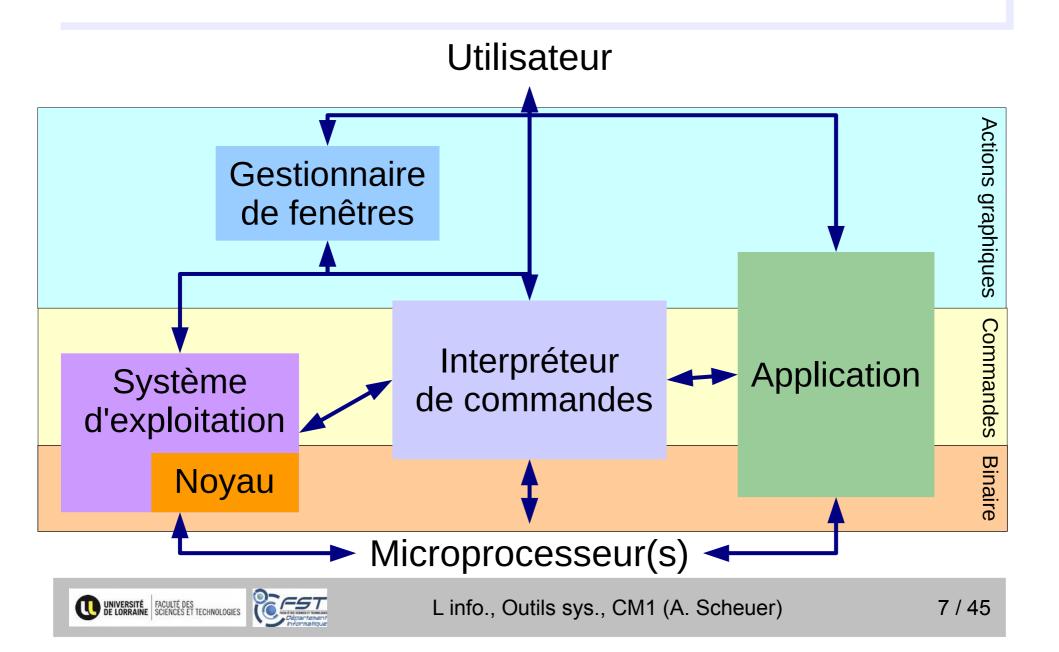
- Interpréteur de commandes bash
 - configuration (.bashrc) et alias
 - manipulations et droits des fichiers
 - makefile
- Traitement de lignes de texte

awk, grep, sed

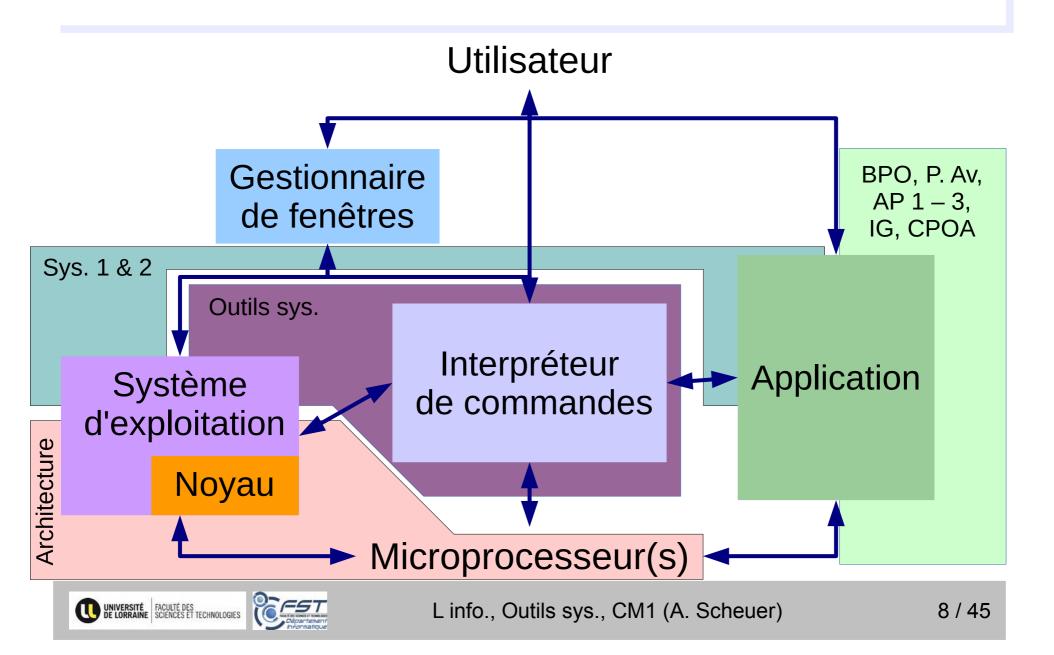
Intérêt?

- Informatique
 - traitement automatique de l'information
- Éviter de réinventer la roue
 - Programmer n'est pas toujours la meilleure solution pour obtenir un traitement
 - Utiliser des outils existants est parfois plus efficace (surtout si les outils le sont)

De l'humain à la machine



Enseignements concernés



Plan

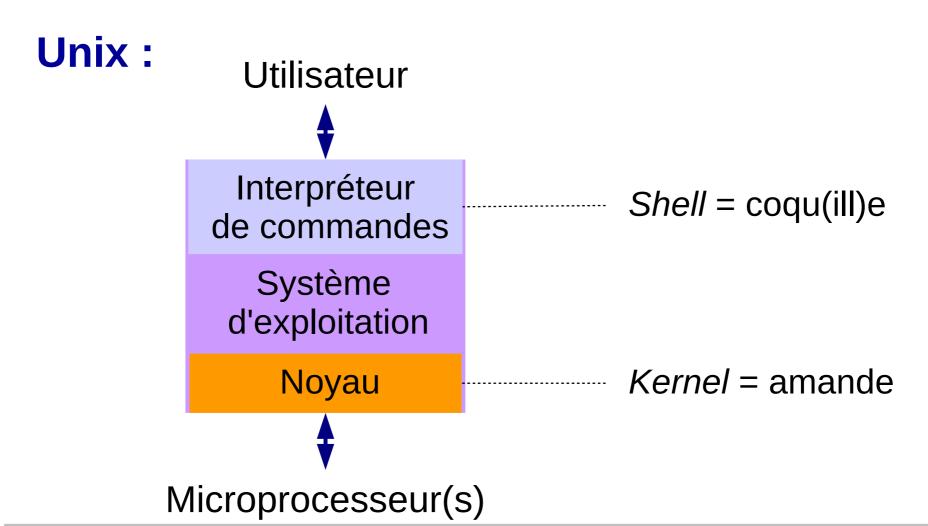
- Généralités
- Interpréteur de commandes
 - Fonctions
 - Spécificités de bash
- Emacs
 - Intérêt par rapport à d'autres éditeurs
 - Commandes simples
 - Aller plus loin

Rôle

Interpréteur de commandes = interface utilisateur du système d'exploitation

- interprète les commandes tapées au clavier (exécute les programmes),
- fournit un langage de programmation interprété,
- permet de modifier l'environnement,
- gère les redirections des entrées et sorties des processus.

Terminologie







Commandes

- Permet de lancer <u>tous les logiciels</u>
 système d'exploitation (droit) + applications
- Propose un manuel pour la plupart
 - Accessible par la commande man [cat] cmd
 - Donne
 - le nom de la commande et son prototype (synopsis),
 - une description de son fonctionnement,
 - ses options et la valeur retournée,
 - des exemples et des références.

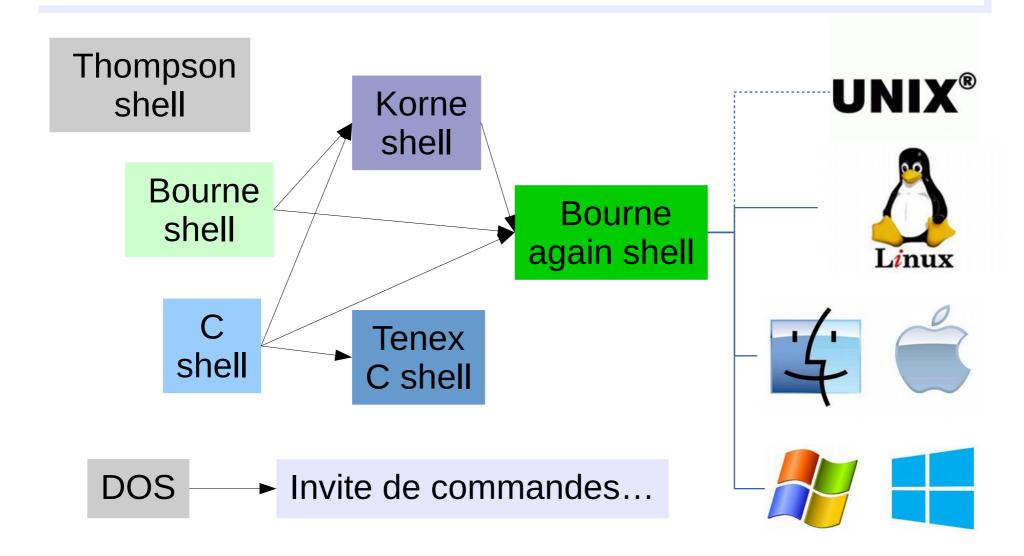




Interpréteur

- Lancer une commande ou un fichier exécutable (séquence de commandes réutilisable)
- Un tel fichier (script) est
 - exécuté ligne à ligne (interprété)
 - comme Python et les .class
 - pas de compilation contrairement au C et à Java
 - écrit dans un langage qui dépend de l'interpréteur

Différents interpréteurs







Différents cadres

- Dans une console
 - Écran texte (non graphique)
 - → lancer le BIOS ou le SE
 - Debian : 6 disponibles
 - (Ctrl + Alt + Fn)
- Dans un terminal
 - Émule la console
 - Grande variété, xterm + simple & + rapide

```
Security (1997) Security (1997)

Security (1997) The Day of Ed. 27 Security and Right

Security (1997) The Day of Ed. 27 Security and Right

Security (1997) The Day of Ed. 27 Security (1997)

- Baseron (1997) The Day of Ed. 27 Security (1997)

- Baseron (1997) The Day of Ed. 27 Security (1997)

Security (1997) The Day of Ed. 27 Security (1997)

Security (1997) The Day of Ed. 27 Security (1997)

Security (1997) The Day of Ed. 27 Security (1997)

Security (1997) The Day of Ed. 27 Security (1997)

Security (1997) The Day of Ed. 27 Security (1997)

Security (1997) The Day of Ed. 27 Security (1997)

Security (1997) The Day of Ed. 27 Security (1997)

Security (1997) The Day of Ed. 27 Security (1997)

Security (1997) The Day of Ed. 27 Security (1997)

Security (1997) The Day of Ed. 27 Security (1997)

Security (1997) The Day of Ed. 27 Security (1997)

Security (1997) The Day of Ed. 27 Security (1997)

Security (1997) The Day of Ed. 27 Security (1997)

Security (1997) The Day of Ed. 27 Security (1997)

Security (1997) The Day of Ed. 27 Security (1997)

Security (1997) The Day of Ed. 27 Security (1997)

Security (1997) The Day of Ed. 27 Security (1997)

Security (1997) The Day of Ed. 27 Security (1997)

Security (1997) The Day of Ed. 27 Security (1997)

Security (1997) The Day of Ed. 27 Security (1997)

Security (1997) The Day of Ed. 27 Security (1997)

Security (1997) The Day of Ed. 27 Security (1997)

Security (1997) The Day of Ed. 27 Security (1997)

Security (1997) The Day of Ed. 27 Security (1997)

Security (1997) The Day of Ed. 27 Security (1997)

Security (1997) The Day of Ed. 27 Security (1997)

Security (1997) The Day of Ed. 27 Security (1997)

Security (1997) The Day of Ed. 27 Security (1997)

Security (1997) The Day of Ed. 27 Security (1997)

Security (1997) The Day of Ed. 27 Security (1997)

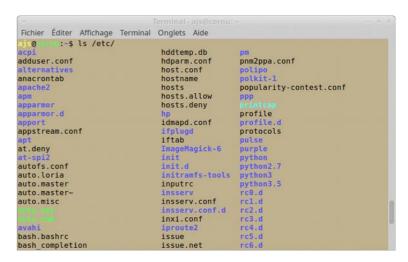
Security (1997) The Day of Ed. 27 Security (1997)

Security (1997) The Day of Ed. 27 Security (1997)

Security (1997) The Day of Ed. 27 Security (1997)

Security (1997) The Day of Ed. 27 Security (1997)

Security (1997)
```





Spécificités de bash

- Complétion (commande, fichier, option, ...)
 très utile pour taper plus vite vos commandes
- Couleurs = + grande lisibilité des résultats
 compilation, ls, appli. (echo/printf + car. sp.)
- Fichiers de configuration
 selon le type du shell (connexion/utilisation)

Cf manuel!





Script: début

La première ligne doit être

#!/bin/bash

 Comme dans de nombreux programmes, des commentaires sont bienvenus (ou même indispensables)

commentaire = toute ligne commençant par #

Variables: définition

Nom de variable = identifiant usuel
 car. alphanumérique et _, pas de chiffre au début

- Valeur affectée avec le caractère égal (=)
 - Pas d'espace autour de =
 - Un seul type : chaîne de caractères

Variables: vérification

- Affichage obtenu avec echo
 - pour vérifier les valeurs de vos variables, ou
 - pour donner des informations de débogage
- Tout texte sur la même ligne = paramètre (données à afficher), chaîne de caractères
- Sans passage à la ligne = -n après echo
- Valeur d'une variable x obtenue avec \$x



Variables : types de chaîne

Guillemets simples → chaîne de caractères

```
echo 'truc pwd $truc' → truc pwd $truc
```

Guillemets doubles → chaîne avec substitution

```
echo "truc pwd $truc" → truc pwd 3
```

si la variable truc vaut 3

Guillemets inversés → exécution

```
echo `truc pwd $truc` →
```

truc : commande introuvable





Variables: calcul

Commencer par let pour faire des calculs (opérations arithmétiques)

$$a=7$$

$$b=2$$

let
$$x=$a+$b$$

$$\rightarrow 9$$



Variables d'environnement

- Variables globales
 - liste obtenue par la commande env
- Une variable est transmise à l'environnement avec la commande export

```
export LESSCHARSET=utf-8
```

Alias : mot → chaîne de caractères

```
alias go=gnomeopen; alias rm='rm -i'
```

Export & alias → .bashrc





Variables: tableau

Définition entre parenthèses

```
tab=(truc bidule chose)
```

- Accès aux éléments avec crochets
 - tab[1]=...
 - echo $\{tab[1]\} \rightarrow bidule$
 - echo \${tab[*]} → truc bidule chose



Condition

```
if [ cond ]; then
...
elif [ cond ]; then
...
else
...
fi
```

Attention aux espaces autour des crochets un point-virgule équivaut à un changt de ligne





Condition: opérateurs logiques

Opérateurs logiques

```
négation : !et : & &ou : | |
```

• Placés entre les conditions (hors crochets)

```
if [ ... ] && ![ ... ]; then ...
```

Condition : chaîne de caractères

Opérateurs

```
- égalité : =
```

- différence : !=

- chaîne vide : -z

- chaîne non vide : -n

Les opérateurs unaires (-...) sont préfixés

```
if [ -z $a ]; then echo "a est vide" fi
```



Condition: entier

Opérateurs de comparaison

- Égalité: -eq; différence: -ne
- Inférieur strict : -lt (lower than) ; inf. ou égal : -le
- Sup. strict: -gt (greater than); sup. ou égal: -ge

Exemple

```
if [ $a -lt $b ]
then echo "a < b"; fi</pre>
```





Condition: fichier

Opérateurs

```
- Existence: -e
```

```
    Fichier: -f; dossier: -d; lien symbolique: -L
```

```
    Lisible: ¬r; modifiable: ¬w; exécutable: ¬x
```

- Plus ancien: -ot (older than); plus récent: -nt

Exemple

```
if [ -f /etc/bash.bashrc ]
then . /etc/bash.bashrc; fi
```





Boucle while

```
while [ cond ]; do
...
done
```

Condition et passages à la ligne similaires aux conditionnelles



Boucles for

Parcours d'une liste

```
for élément in liste ; do ...; done
```

Itérations classiques

```
for (( expr1 ; expr2 ; expr3 ))
do ...; done
```

Exemple

```
for fich in `ls`; do if [ -x $fich ] then `./$fich`; fi; done
```



Fonction

Définition

- maFonction() { ... }
- function maFonction { ... }
- Paramètres : accès par \$1, \$2, ...; nombre = \$#

Appel

- maFonction param1 param2 ...
- retour de la dernière fonction appelée = \$?



Compléments

Plus de détails dans le manuel (compliqué !!! Présentation simplifiée !)



Plan

- Généralités
- Interpréteur de commandes
 - Fonctions
 - Spécificités de bash
- Emacs
 - Intérêt par rapport à d'autres éditeurs
 - Commandes simples
 - Aller plus loin





Éditer sans interface graphique

- Nécessaire dans certains contextes
 - Serveurs : périphériques absents, IG = gaspi
 - Systèmes embarqués : peu ou pas d'IG
 - Aéronautique, automobile
 - Industrie
 - Montre connectée, petit téléphone, ...
- Seulement possible avec quelques logiciels
 - Nano : simple et facile d'utilisation mais limité
 - Vi/vim ou emacs : plus compliqués mais puissants





Emacs: intérêt

- Très puissant, peu exigeant
- Reconnaissance syntaxique
 - texte en couleur + indentation intelligente + fonctions dépendant du type de document
- Fenêtre de commande intégrée
 - exécution d'une commande ou d'un script
 - gestion de l'affichage (couleur + liens)
- Complètement configurable





Emacs: peu exigeant

Avec ou sans environnement graphique:

- dans une fenêtre spécifique (défaut), ou dans la fenêtre de commande (option -nw)

emacs@adescartes Edit Options Buffers Tools C++ Help // This class needs the definition of iSeeML::rob::CompPath Winclude <iSeeML/rob/CompoundPath.hpp> // This class needs the definition of iSeeML::rob::LinCurvPath Editer Affichage Terminal Ongels Akte Winclude <iSeeML/rob/LinCurvPath.hpp> ile Edit Options Buffers Tools C++ Help /** This class defines Dubins like paths, which are similar ** to @ref iSeeML::rob::DubinsPath *Dubins paths". ** A Dubins like path is made of at most three parts, which can be This class defines Dubins like paths, which as ** a turn or a straight line. ** @ref iSeeML::rob::DubinsPath "Dubins paths" parts are circular ** arcs or line segments, with a ponctual change of curvature " a turn or a straight line.
" gred (SeeMI::rob::DubinsPalh *Dubins palhs*); ** at each part's end, while @ref 15eeML::rob::FscPath "FSC paths" ** turns start and finish with zero curvature. arcs or line segments, with a ponetual change at each part's end, while gire (SeeMillions:) ** Each part is represented by one to three pieces, which are ** turns start and finish with zero curvature.
** Lach part is represented by one to three preci-** @ref iSeeML::rob::LinCurvPath 'Linear curvature paths' ** @ref iSeeML::rob::LinGurvPath *linear curvatur ** @author @ref Aj5 "Alexis Scheuer" class iSeeML::rob::DubinsLikePath : public iSeeML::rob::CompoundPath // === Inner Type Definitions public: Dubins-like paths can be of six types: their /+* Dubins-like paths can be of six types: their first and last ** parts are turns, their middle part is tangent to the first ** and last parts but can be a line segment or a turn. Thus, Dubins-like paths' types can be named ** Thus, Dubins-like paths' types can be named by three "" letters: the first and third are #s 1 or #s ** letters: the first and third are @a l or @a r (Left or Right turn) and the second is @a 1, @a r or @a s DubinsLikePath.hpp 2% (22,31)

avec la souris active partout (défaut),
 partiellement active (dans la fenêtre de commande)
 ou sans (tout peut être fait au clavier!)





Emacs: pratique

- Complétion (touche de tabulation)
 - commande
 - répertoire et fichier
- Visualisation
 - fenêtre de commande (⇒ navigation dossiers & fichiers)
 - images (GIF, JPEG, PNG, ..., SVG, ...)
 - documents (PS, PDF)



Emacs pour programmer

- Couleurs (mots clé, types, variables, fonctions)
- Indentation intelligente
 - fonctionnalité rare (aussi dans eclipse)
 - détecte les erreurs avant interprétation/compilation
- Fenêtre de commandes: M-x shell
- Compilation intégrée
 commande au choix (make -k par défaut)



Emacs pour tout

Configuration dans des fichiers :

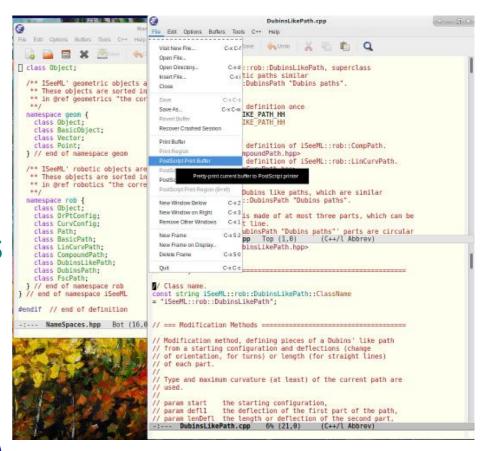
- couleurs & comportement
- syntaxe et fonctions d'un nouveau langage
- configuration en Lisp
 (.el = Emacs Lisp)

```
emacs@adescartes
File Edit Options Buffers Tools Emacs-Lisp Help
                    Save 衡 Undo 💥 📋 🛍 🔾
 (defun langB-mode ()
   "Major mode for editing Machine files (in B Language)."
   (interactive)
   (setq major-mode 'langB-mode)
   (setq mode-name "B Language")
   (font-lock-add-keywords
    '(("\\<\\(MODEL\\|MACHINE\\|REFINEMENT\\|IMPLEMENTATION\\|REFINES\\|CONSTRAI
snts\\|Definitions\\|Invariant\\|Sets\\|Constants\\|Concrete_constants\\|Abstract*
CONSTANTS\\|HIDDEN CONSTANTS\\|VISIBLE CONSTANTS\\|PROPERTIES\\|VARIABLES\\|CON>
⊈CRETE VARIABLES\\|ABSTRACT VARIABLES\\|HIDDEN VARIABLES\\|VISIBLE VARIABLES\\|IN•
$\VARIANT\\|VALUES\\|ASSERTIONS\\|INITIALISATION\\|OPERATIONS\\|PROMOTES\\|SEES\\|P
("/\\*" "\\*/" font-lock-comment-face)
       ("\\<\\(TRUE\\|FALSE\\)\\>" . font-lock-constant-face)
        ("" . font-lock-doc-face)
         ("" . font-lock-function-name-face)
       ("\\<\\(BEGIN\\|PRE\\|END\\|IF\\|THEN\\|ELSE\\|SELECT\\|WHEN\\|WHILE\\|DO\\
-:--- langB-mode.el 9% (15,240) (Emacs-Lisp)
(defvar scilab-unterminated-string-face 'scilab-unterminated-string-face
   "Self reference for unterminated string face.")
 (defvar scilab-scicos-keyword-face 'scilab-scicos-keyword-face
   "Self reference for scicos keywords.")
 (defvar scilab-valid-variable-name "\\<[A-Za-z$# %][A-Za-z0-9$# ]*\\>"
   "Regexp describing all valid variable names")
 (defun scilab-simple-send (proc string)
  "Default function for sending in scilab session to PROC input STRING.
 Takes in account the meaning of the symbol '!'.
 See the hook `comint-input-sender'."
 -:--- scilab.el
                      10% (635,0)
                                    (Emacs-Lisp)
```



Emacs: bases

- Édition d'un fichier dans un tampon (buffer)
 - +ieurs tampons, dans la même fenêtre ou une autre
- Flèches, souris et menus
- Raccourcis claviers & commandes
 - pratique & lecture
 du didacticiel (Ctrl-h t)

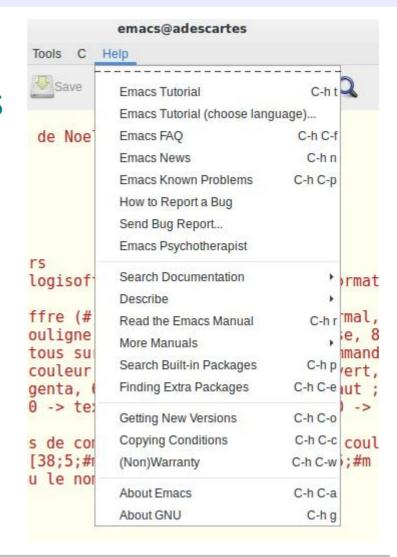




Emacs: premiers pas

- Commandes mnémoniques
 Mouvements, fichiers
- Annulation récursive
 liste d'actions effectuées toujours croissante
- Aide abondante

Ctrl-h .../menus





Emacs: aller plus vite

Fonctions insérant du texte

- Shell-script : beaucoup de choses (ajout, exéc., ...)
- LaTeX: insertion de bloc, fin de bloc, ...
 latex-insert-block, latex-close-block, ...
- Doxygen: insertion de commentaire (fonction, ...)
 doxymacs-insert-function-comment, doxymacs-insert-file-comment, ...
- C / C++: rien actuellement (à ajouter ? Classe, ...)

Emacs: configurer

- Fichiers de configuration accessibles
- Exemples, wiki et manuels en ligne
- Toujours garder un emacs ouvert lors des tests ouvrir un autre pour vérifier

Emacs: références

- Introduction (Ubuntu, wikipedia)
- GNU Emacs Manual (en)
- Wiki Emacs (fr / en)
- Emacs ou VI?



La semaine prochaine

- Cours mardi à 16 h 15
- TP 1 (date et heure selon votre groupe)
 - vous aurez besoin de bash
 et d'un éditeur (emacs de préférence)
 - solution simple : ajouter emacs (téléchargé ici)
 - plus complet :
 - installer une machine virtuelle avec Ubuntu
 - sous Windows 10, utiliser WSL