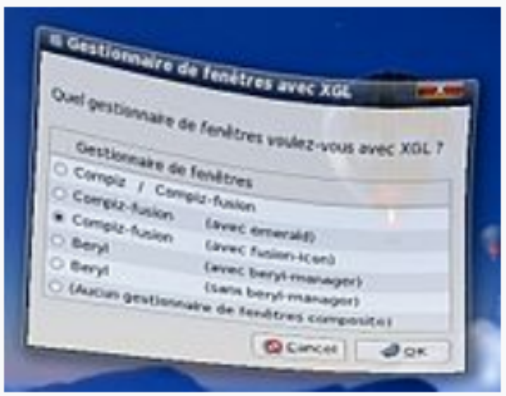
Exemples effets graphiques : Le gestionnaire fenêtres Compiz

- Spectaculaire mais consommation importante de ressources

l'effet « flammes »

l'effet « lampe magique »

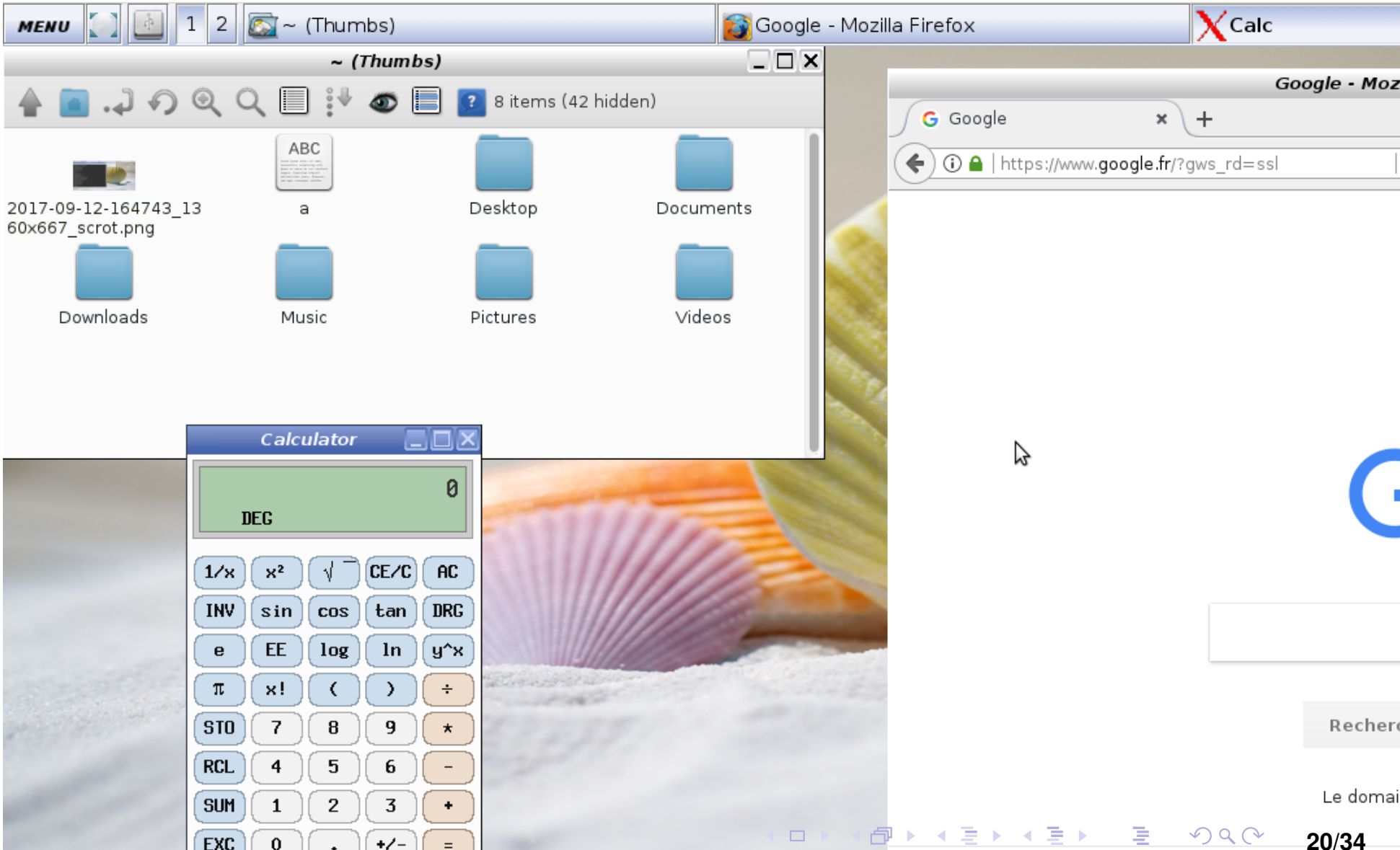
bureau sur un cube.



effet fenêtre molle.

Le gestionnaire de fenêtres IceWM

- utilisé par défaut sous `Antix Linux`
- Pas spectaculaire mais très léger et rapide

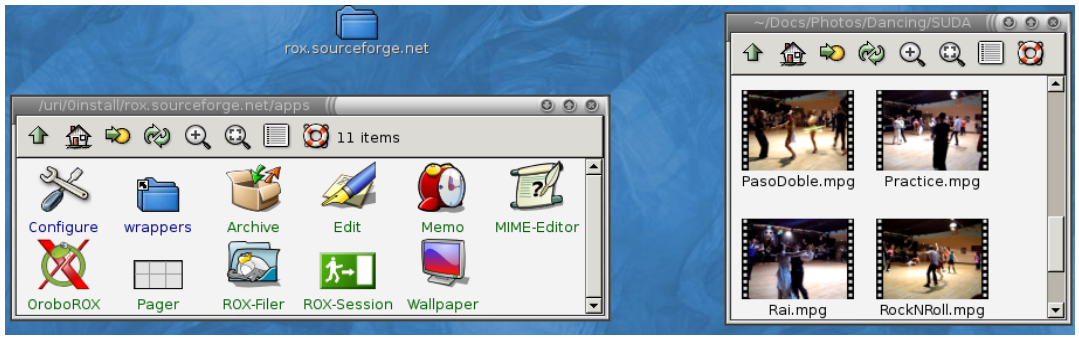


Exemple de systèmes légers/rapides

distribution : Antix (très complet)

noyau : Linux

interface graphique Rox (remplaçable)



gestionnaire fenêtres IceWm (remplaçable)

$\Longrightarrow$ Ne pas jeter votre vieux ordinateur,

$\Longrightarrow$ Sortir de l«l'obsolescence programmee»

TinyCoreLinux est encore plus petit et il y en a d'autres...

Facile à lancer dans `virtualbox` pour tester

Terminal Shell : gestion de dossiers/fichiers

Notion de dossier courant

- Le nommage des fichiers peut se faire par rapport à un dossier courant.
- On peut l'afficher avec la commande `pwd`

Rappel nommage :

```
« .. » = dossier parent/père
« / »  = dossier racine du système de fichiers
« ~ »  = dossier personnel ($HOME)
```

Terminal Shell : gestion de dossiers/fichiers

Commandes qui manipulent des dossiers et fichiers

`cd /`	se placer à la racine
`cd /usr/games`	aller au dossier «/usr/games»
`cd ..`	se placer dans le dossier parent
`cd games`	aller au dossier «games» du dossier courant
`ls`	afficher tous les fichiers du dossier courant
`ls f*`	afficher les fichiers qui commencent avec `f`
`./fortune`	lancer le programme « fortune » du dossier courant
`cd ~`	aller au dossier personnel ($HOME)

Rappel nommage :

```
« .. » = dossier parent/père
« / »  = dossier racine du système de fichiers
« ~ »  = dossier personnel ($HOME)
```

Terminal Shell : gestion de dossiers/fichiers

```
daniel@dcnam ~ $ cd /
daniel@dcnam / $ cd /usr/games/
daniel@dcnam /usr/games $ cd ..
daniel@dcnam /usr $ cd games
daniel@dcnam /usr/games $ ./fortune
Be careful of reading health books, you might die of a misprint.
                -- Mark Twain
daniel@dcnam /usr/games $ cd ~
daniel@dcnam ~ $ pwd
/home/daniel
daniel@dcnam ~ $
```

Rappel nommage :

« .. » = dossier parent/père
« / » = dossier racine du système de fichiers
« ~ » = dossier personnel ($HOME)

La mémoire RAM sous Linux reporté

- Le noyau peut réserver une zone de mémoire à chaque application
  - le noyau protège contre les accès illégaux des applications vis à vis du noyau : impossible d'écrire sur la mémoire d'un autre programme ou dans le *kernel space*
- Mémoire Tampon (Buffers) et Cache : stocker dans la RAM une part du disque dur pour accélérer la machine
- SWAP : un fichier d'échange dit « swap » est utilisé lorsque la RAM est insuffisante (mémoire virtuelle)
  - Linux : faire une partition SWAP de même taille que la RAM

Disque(s) dur(s)

- $\pm$ 50 fois moins rapides que la RAM
- Racine unique « `/` », pas de lecteur « `C:` » Windows
- Les clés USB, les DVD sont représentés par un fichier comme `/dev/sdb`, `/dev/sdc` qui doit être monté sur un

La mémoire RAM sous Linux reporté

- Le noyau peut réserver une zone de mémoire à chaque application
  - le noyau protège contre les accès illégaux des applications vis à vis du noyau : impossible d'écrire sur la mémoire d'un autre programme ou dans le *kernel space*
- Mémoire Tampon (Buffers) et Cache : stocker dans la RAM une part du disque dur pour accélérer la machine
- SWAP : un fichier d'échange dit « swap » est utilisé lorsque la RAM est insuffisante (mémoire virtuelle)
  - Linux : faire une partition SWAP de même taille que la RAM

Disque(s) dur(s)

- $\pm$ 50 fois moins rapides que la RAM
- Racine unique « `/` », pas de lecteur « `C:` » Windows
- Les clés USB, les DVD sont représentés par un fichier comme `/dev/sdb`, `/dev/sdc` qui doit être monté sur un

La mémoire RAM sous Linux reporté

- Le noyau peut réserver une zone de mémoire à chaque application
  - le noyau protège contre les accès illégaux des applications vis à vis du noyau : impossible d'écrire sur la mémoire d'un autre programme ou dans le *kernel space*
- Mémoire Tampon (Buffers) et Cache : stocker dans la RAM une part du disque dur pour accélérer la machine
- SWAP : un fichier d'échange dit « swap » est utilisé lorsque la RAM est insuffisante (mémoire virtuelle)
  - Linux : faire une partition SWAP de même taille que la RAM

Disque(s) dur(s)

- $\pm$ 50 fois moins rapides que la RAM
- Racine unique « `/` », pas de lecteur « `C:` » Windows
- Les clés USB, les DVD sont représentés par un fichier comme `/dev/sdb`, `/dev/sdc` qui doit être monté sur un

Visualiser l'état des mémoires à l'aide du Shell

reporté

Démarrer une console/terminal et taper les commandes indiquées à gauche (explication à droite en commentaire) :

```
free                    #informations sur la RAM
free -h                 #h = human readable
cat /proc/meminfo
top                     #taper M pour trier selon
                        #la consommation memoire
df -h                   #informations disques durs
```

Pour visualiser les clés USB montées :

```
cat /proc/mounts
mount
```

```
daniel@dcnam ~ $ mount
/dev/sda1 on / type ext4 (rw,errors=remount-ro)
/dev/sdb1 on /media/daniel/Barcelon type vfat (rw,nosuid,node
```

clé USB

Gérer d'autres ressources à l'aide du `Shell`

```
cat /proc/cpuinfo    #Informations CPU
cat /proc/version
uname --help         #affiche les options =>
uname -srv           #Kernel name, version, OS
```

Gérer un processus (ex. `firefox`) s'il consomme trop de CPU

```
killall -SIGSTOP firefox  #firefox suspendu
...
killall -SIGCONT firefox  #firefox redémarre
cpulimit -e firefox -l 50 #utiliser max 50% CPU
```

Arrêter un processus de manière forcé :

```
sudo killall -SIGKILL firefox
```

sudo permet de passer en mode super-utilisateur (root), mot de passe demandé

Gérer d'autres ressources à l'aide du Shell

```
cat /proc/cpuinfo   #Informations CPU
cat /proc/version
uname --help        #affiche les options =>
uname -srv          #Kernel name, version, OS
```

Gérer un processus (ex. `firefox`) s'il consomme trop de CPU

```
killall -SIGSTOP firefox  #firefox suspendu
...
killall -SIGCONT firefox  #firefox redémarre
cpulimit -e firefox -l 50 #utiliser max 50% CPU
```

Arrêter un processus de manière forcé :

```
sudo killall -SIGKILL firefox
```

sudo permet de passer en mode super-utilisateur (root), mot de passe demandé

Quelques commandes usuelles

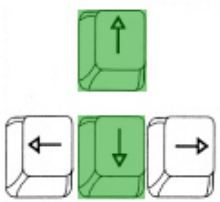
Terminal

```
daniel@dcnam ~ $ cd /tmp/                          aller dans le dossier "/tmp/"
daniel@dcnam /tmp $ mkdir vari1                    créer le dossier "vari1"
daniel@dcnam /tmp $ cd vari1                       aller dans le dossier "vari1"
daniel@dcnam /tmp/vari1 $ echo "Salut"             afficher un message
Salut
daniel@dcnam /tmp/vari1 $ echo "Salut">test.txt    écrire "Salut" dans  un fichier
daniel@dcnam /tmp/vari1 $ cat test.txt             afficher le fichier
Salut
daniel@dcnam /tmp/vari1 $ cp test.txt test2.txt    copier test.txt en test2.txt
daniel@dcnam /tmp/vari1 $ ls                       lister fichiers dossier courant
test2.txt  test.txt
daniel@dcnam /tmp/vari1 $ mv test.txt test3.txt    déplacer/renommer fichiers
daniel@dcnam /tmp/vari1 $ ls
test2.txt  test3.txt
daniel@dcnam /tmp/vari1 $ rm test3.txt             effacer le fichier test3.txt
daniel@dcnam /tmp/vari1 $ ls
test2.txt
daniel@dcnam /tmp/vari1 $ ls -l                    lister fichiers format  long
total 4 propriétaire          taille fichier (octets)
-rw-r--r-- 1 daniel daniel 6 Sep 29 16:21 test2.txt
daniel@dcnam /tmp/vari1 $ cd ..                    revenir au dossier parent
daniel@dcnam /tmp $ rm -r vari1                    effacer le dossier "vari1"
```

Touches très utiles : [Tab], flèches ↑↓, CTRL

Tapez le début d'une commande/fichier ⊕ appuyer sur la touche TAB : Linux complète la saisie ou propose les différentes possibilités pour la compléter

```
daniel@dcnam ~ $ echo "salut" > aadqlskdjqskdjqslkd
daniel@dcnam ~ $ cat aadqlskdjqskdjqslkdjqskldjqsd
                   taper aa + [TAB] et linux remplit le reste!
```



Les flèches haut et bas permettent de revenir sur les commandes tapées récemment et de naviguer sur ces commandes

CTRL-C permet d'arrêter une commande (programme) lancée

CTRL-Z envoye la commande lancée en arrière-plan (taper `fg` pour revenir)

Le manuel

Il est disponible si on tape `man` suivie de la commande qu'on veut étudier

- `man ls` : le manuel de la commande `ls` (lister fichiers)
- `man cat` : le manuel de la commande `cat` (afficher fichiers)

Pour naviguer dans la documentation :

- [`ESPACE`] : page suivante
- `CTRL-u` : page précédente
- `/` : chercher un mot clé
- `n` : aller à la prochaine apparition du mot clé

Redirection de la sortie standard reporté

Fonctionnement classique :

- on tape une commande (ex, `ls, cat`);
- le résultat/sortie s'affiche dans le terminal actif

Il est possible de rediriger ce résultat :

`ls > fic.txt`	→	écrire le résultat dans le fichier `fic.txt` (écraser le contenu précédent)
`ls >> fic.txt`	→	ajouter le résultat au fichier `fic.txt` (sans rien écraser)
`ls -l /etc/| more`	→	lançer `ls -l /etc/` et envoyer le résultat à la commande `more`, pour l'afficher page par page

Redirection de la sortie standard reporté

Fonctionnement classique :

- on tape une commande (ex, `ls, cat`);
- le résultat/sortie s'affiche dans le terminal actif

Il est possible de rediriger ce résultat :

`ls > fic.txt`	→	écrire le résultat dans le fichier `fic.txt` (écraser le contenu précédent)
`ls >> fic.txt`	→	ajouter le résultat au fichier `fic.txt` (sans rien écraser)
`ls -l /etc/| more`	→	lançer `ls -l /etc/` et envoyer le résultat à la commande `more`, pour l'afficher page par page

En plus de la sortie, tout programme possède une entrée et une sortie d'erreurs généralement associées au terminal actif.

reporté

En plus de la sortie, tout programme possède une entrée et une sortie d'erreurs généralement associées au terminal actif. reporté

`ls 2> err.txt` → redirection des erreurs vers `err.txt`
`irb<<<"2+3"` → la commande `irb` reçoit `"2+3"` comme entrée

Exemple : comparer les deux commandes suivantes

```
find / -name "virtual"
find / -name "virtual" 2>/dev/null
```

- /dev/null=nulle part

En plus de la sortie, tout programme possède une entrée et une sortie d'erreurs généralement associées au terminal actif. reporté

`ls 2> err.txt` → redirection des erreurs vers `err.txt`
`irb<<<"2+3"` → la commande `irb` reçoit `"2+3"` comme entrée

Exemple : comparer les deux commandes suivantes

```
find / -name "virtual"
find / -name "virtual" 2>/dev/null
```

- /dev/null=nulle part

Commandes `grep`, `wc`, `sed`, `cut` reporté

Questions/exercices :

1. Afficher les lignes qui comportent le mot «toto» dans un fichier donné.
2. Compter le nombre de ces lignes
3. Afficher que les adresses mail `gmail` dans un fichier
4. Remplacer «toto» avec «dodo»
5. Afficher que les premiers dix colonnes du résultat d'une commande

Commandes `grep`, `wc`, `sed`, `cut` reporté

Questions/exercices :

1. Afficher les lignes qui comportent le mot «toto» dans un fichier donné.
2. Compter le nombre de ces lignes
3. Afficher que les adresses mail `gmail` dans un fichier
4. Remplacer «toto» avec «dodo»
5. Afficher que les premiers dix colonnes du résultat d'une commande

Un programme compilé est une “commande”

reporté

La syntaxe du lancement du programme exécutable est similaire à une commande :

exemple : `./monprogramme argument(s)`

`monprogramme` pourrait être compilé à partir de C ou C++ ou Pascal ou tout autre langage

Un programme C tout petit : on donne la définition d'une fonction main() qui est exécutée en premier. Elle affiche «Salut» :

```
void main(){
    printf("Salut");
}
```

Compilation : `gcc main.c -o executb`
Exécution : `./executb`

gcc=compilateur C,
main.c=fichier C
(.c pas .cpp!)

Un programme compilé est une “commande”

reporté

La syntaxe du lancement du programme exécutable est similaire à une commande :

exemple : `./monprogramme argument(s)`

`monprogramme` pourrait être compilé à partir de C ou C++ ou Pascal ou tout autre langage

Un programme C tout petit : on donne la définition d'une fonction main() qui est exécutée en premier. Elle affiche «Salut» :

```
void main(){
    printf("Salut");
}
```

Compilation : `gcc main.c -o executb`
Exécution : `./executb`

`gcc`=compilateur C, `main.c`=fichier C (`.c` pas `.cpp`!)

Un programme compilé est une “commande”

reporté

La syntaxe du lancement du programme exécutable est similaire à une commande :

exemple : `./monprogramme argument(s)`

`monprogramme` pourrait être compilé à partir de C ou C++ ou Pascal ou tout autre langage

Un programme C tout petit : on donne la définition d'une fonction main() qui est exécutée en premier. Elle affiche «Salut» :

```
void main(){
    printf("Salut");
}
```

Compilation : `gcc main.c -o executb`
Exécution : `./executb`

`gcc`=compilateur C, `main.c`=fichier C (`.c` pas `.cpp`!)

Calculer le cube d'un argument en C reporté

la fonction main() renvoie void=vide/rien

argument du programme/commande

```c
void main(int argc, char** argv){
    int x,cube ;          //declaration variables
    x = atoi(argv[1]); //conversion vers entier
    cube = x*x*x;
    printf("%d", cube);
}
```

- Tous les systèmes qu'on a vu sont écrits en C !
- Il est quasiment irremplaçable, il date des années 1970
- Il a influencé `C++` et `Java` et par la suite `Processing`

  $\Longrightarrow$ `Processing` utilise une syntaxe de type `C/C++`

Calculer le cube d'un argument en C reporté

la fonction main() renvoie void=vide/rien

argument du programme/commande

```c
void main(int argc, char** argv){
    int x,cube ;          //declaration variables
    x = atoi(argv[1]); //conversion vers entier
    cube = x*x*x;
    printf("%d", cube);
}
```

- Tous les systèmes qu'on a vu sont écrits en C !
- Il est quasiment irremplaçable, il date des années 1970
- Il a influencé `C++` et `Java` et par la suite `Processing`

  $\Longrightarrow$ `Processing` utilise une syntaxe de type `C/C++`