ATELIER UNIX/LINUX S2-APP3

Département de génie électrique et de génie informatique

Université de Sherbrooke

3 février 2015

OBJECTIFS

- Faire les premiers pas avec UNIX/LINUX
- Développer des logiciels en C++ avec g++ et make

RÉFÉRENCES UTILES

- Petit guide UNIX (sur le site de l'APP)
- Livre de C++
- Répertoire sur le serveur: présentation + code
  /grp/cours/s2gen241/Atelier
- Ressources WEB
  (non accessible lors de l'examen)

ÉTAPES PROPOSÉES

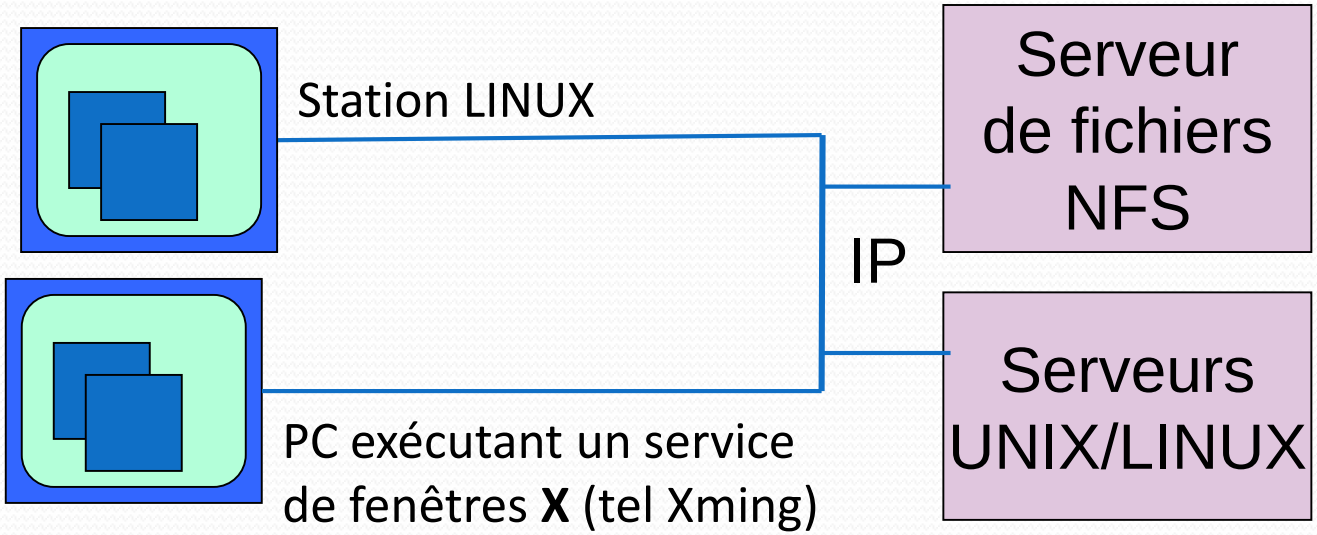
- Survol rapide de LINUX/UNIX
- Processus de compilation
- Étape 1: démarrer une session Linux au laboratoire
- Étape 2: ouverture de session *login* et *logout*
- Étape 3: exploration de l'environnement graphique
- Étape 4: travail en ligne de commande
- Étape 5: gestion de fichiers et de dossiers (console)
- Étape 6: compilation
- Étape 7: compilation séparée et *makefile*
- Étape 8: utilisation du débogueur
- Étape 9: archiver un projet

SURVOL RAPIDE DE LINUX/UNIX

- UNIX créé dans les années 60 (labos ATT) sur des machines PDP-7 et PDP-11
- Le langage C fait partie intégrante de UNIX
- Système multitâches et multi-utilisateurs
- LINUX : implémentation libre de UNIX
- UNIX et LINUX très répandus:
  - Stations de travail
  - Systèmes embarqués et réseautique
  - Serveurs
- C'est aussi le système d'exploitation de base pour MAC-OS (UNIX BSD)

UNIX: SYSTÈME DISTRIBUÉ

- Les fichiers des comptes d'utilisateurs peuvent résider sur un serveur NFS
- Des programmes (applications) peuvent être exécutés sur des serveurs d'application éloignés.

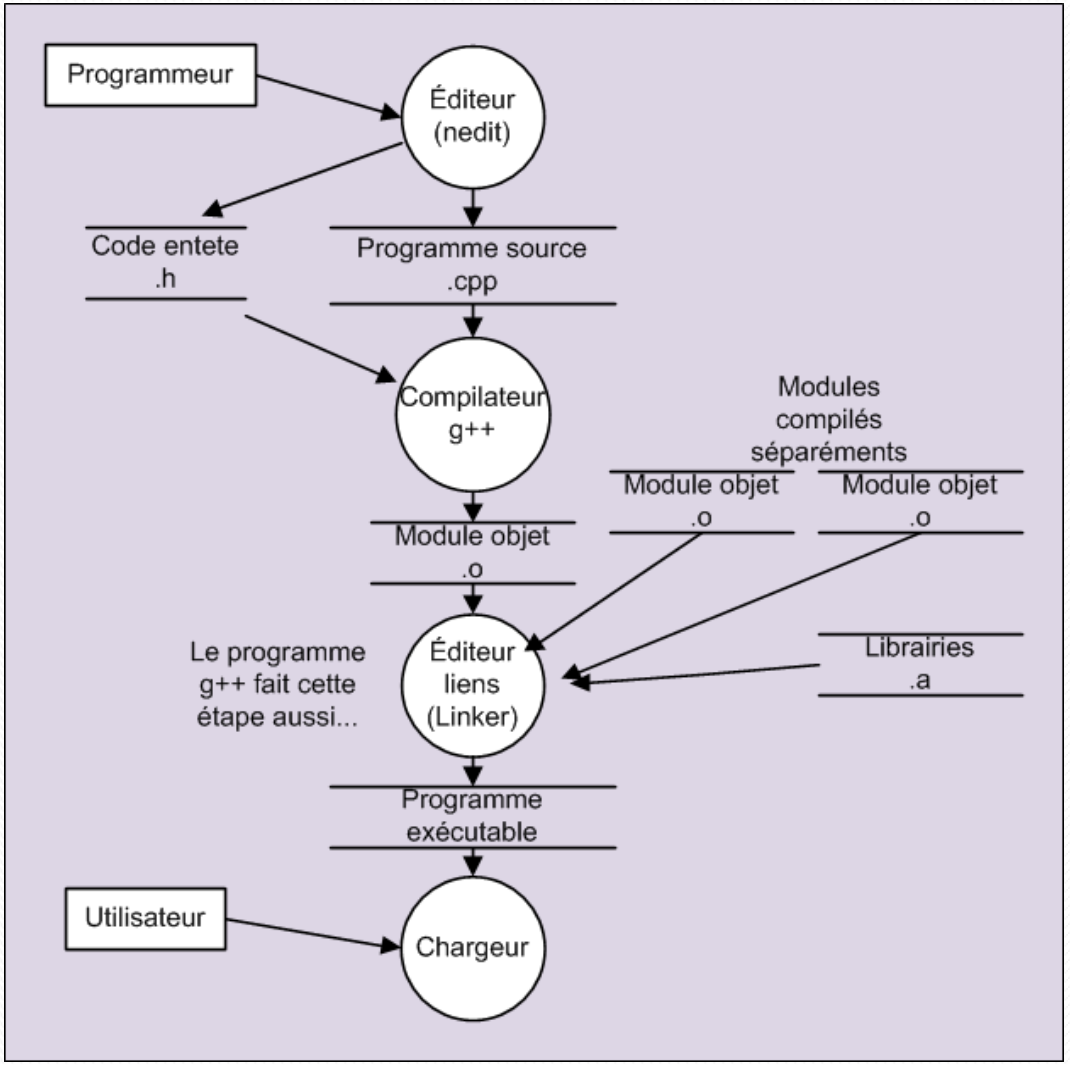


UNIX: SYSTÈME DE FICHIERS

- Système de fichiers hiérarchisé et distribué accessible via le réseau (NFS)
- La localisation d'un système de fichier sur NFS est transparente: tout se passe comme si les fichiers sur un serveur NFS résidaient sur la station locale.
- Mécanismes de protection
- Concept de groupes
- Description de chemins absolus, relatifs: exemple
  /grp/cours/s2gen241
  ../../tmp/data.dat

PROCESSUS DE COMPILATION

Compilation séparée



ÉTAPE 1:démarrer une session Linux

- Dans l'un des laboratoires du Département: redémarrer une station et à l'étape d'amorçage (dans les 10 premières secondes), choisir CentOS.
- Un dialogue d'ouverture de session apparaît (voir la page suivante).
- L'amorçage se fait et on obtient un bureau graphique à l'écran.
- Votre système de fichier personnel est un système résidant sur le réseau NFS départemental.

ÉTAPE 2: *login* et *logout*

- Ouverture de session sur une station (login)
  - Identificateur: votre CIP
  - Mot de passe : le mot de passe de votre courriel de l'université
- Fermeture de session: par le menu de la station
- Ouverture de session distante par un terminal: utiliser un client SSH pour établir une connexion distante via un terminal.
- La commande `exit` déconnecte un terminal de commande.

ÉTAPE 3: environnement graphique

- Explorer l'environnement graphique pour utiliser les navigateurs de fichiers et exécuter les programmes usuellement utilisés sur une station de développement.
- Dans la fenêtre du navigateur de fichiers: si le dossier `/grp` ne paraît pas: taper le chemin `/grp/cours/s2gen241` dans l'espace qui apparaitra après cliquer « Aller à », « Emplacement ».
- Localiser le programme qui fait apparaître un terminal de commande. Le reste de l'atelier se fera essentiellement à partir des commandes données sur un terminal de commande.
- Ces commandes sont interprétées par un *shell* de commande.

ÉTAPE 4: ligne de commande (terminal)

- Un *shell* de commande est une interface de base. On donne les commandes en réponse à une chaîne de caractères de sollicitation (*prompt*)
- Exemples de shell: *c shell*, *tcsh*, *bash*
- Quelques caractères spéciaux

Caractère	Fonction
`tab`	Compléter
`ctrl-d`	Fin de fichier
`ctrl-c`	Interruption de programme
`&`	Exécution en arrière plan
`>` et `<`	Redirection entrée/sortie: < entrée et > sortie
`~`	Représente le répertoire *home*
`.` et `..`	Le répertoire courant et le répertoire parent

ÉTAPE 4: diverses commandes

Commande	Fonction
`date`	Afficher date courante
`whoami`	Afficher le nom de l'utilisateur
`echo $SHELL`	Affiche le nom du *shell* actif
`ps`	Lister des processus actifs
`cat xyz`	Lister le fichier *xyz*, voir aussi la commande *more*
`gedit &`	Invoquer éditeur *gedit*
`gedit xyz &`	Invoquer éditeur *gedit* et ouvrir le fichier *xyz*

- L'éditeur `gedit` est recommandé.
- Il existe aussi d'autres éditeurs, comme `vi`.
- Notez le rôle du `&` pour revenir à la ligne de commande sans attendre la fin du programme

ÉTAPE 4: autres commandes

Commande	Fonction
`g++`	Invoquer le compilateur g++
`make`	Invoquer l'utilitaire make
`ddd tst &`	Exécuter et débuguer le programme tst
`man xyz`	Consulter le manuel (man pages) sur la commande xyz
`vi`	Invoquer l'éditeur de texte vi, un éditeur en mode console

`vi` est un ancien éditeur encore utilisé, car il fonctionne dans le terminal sans interface graphique. Pour sortir de `vi` rapidement: shift-zz

ÉTAPE 5: gestion fichiers/dossiers

Commande	Fonction
`pwd`	Afficher le répertoire courant
`cd xyz`	Changer (descendre) dans le sous- répertoire xyz
`cd ..`	Remonter dans le répertoire parent
`ls`	Lister le contenu du dossier courant
`ls -l`	Lister avec détails le contenu du dossier courant
`rm xyz`	Détruire le fichier xyz
`rmdir xyz`	Détruire le répertoire (vide) xyz
`mkdir xyz`	Créer un répertoire nommé xyz
`cp`	Copier un fichier
`mv`	Déplacer ou renommer un fichier

ÉTAPE 6: compiler un programme

Dans votre compte (vos dossiers!)

- Vérifier la localisation: `pwd`
- Créer, au besoin, un répertoire : `mkdir Atelier`
- Aller dans ce répertoire: `cd Atelier`
- Copier le programme pr10-3.cpp:
  `cp /grp/cours/s2gen241/Atelier/pr10-3.cpp .`
  Ne pas oublier le . à la fin (répertoire courant)
- Lister, pour vérifier: `ls -ls`
- Compiler (la manière la plus simple): `g++ pr10-3.cpp`
- Lister pour voir les fichiers créés: `ls -lat`
- Exécuter : `./a.out`

ÉTAPE 6: notes

- Dans cet atelier, on présente les commandes telles qu'appliquées dans un terminal de ligne de commande.
- Dans l'interface usager graphique, on peut manipuler les fichiers avec le *drag & drop* et on peut démarrer un éditeur en cliquant un document
- Si on est limité à une interface en ligne de commande (avec client SSH par exemple), au lieu de `gedit`, on utilise éventuellement l'éditeur `vi`

ÉTAPE 7: compilation et *makefile*

- Un `makefile` est un fichier texte qui contient des règles qui permettent de gérer un projet de compilation, entre autres.
- Les règles ont la forme suivante:

```
cible: liste des dépendances
    liste de commandes
```

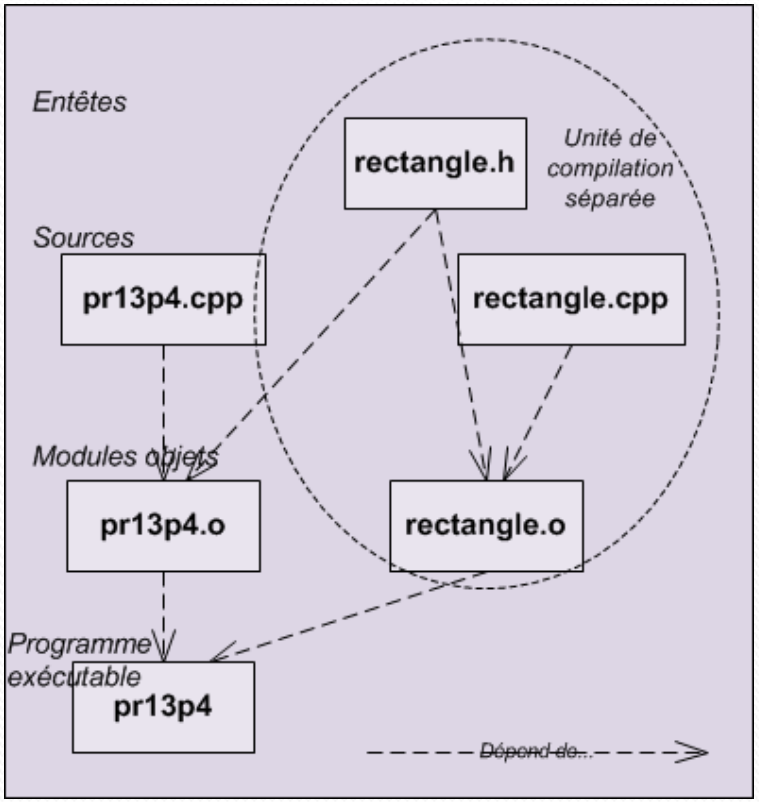
- Exemple de règle:

```
A: B C D
    cp X A
```

- Signification: la cible `A` est à jour, si les dépendances `B`, `C` et `D` sont à jour. Si une ou plusieurs des dépendances doivent être reconstruites, alors il faut reconstruire `A`. Pour reconstruire A, il faut faire la commande `cp X A`.

ÉTAPE 7: compilation et *makefile*

Diagramme de dépendances



ESSENTIEL: caractère TAB et non pas des espaces!!!

```
# makefile No 1 du programme P13p4,
# compilation separee simple.
# But principal:
#   pr13p4 construire programme pr13p4
# Buts secondaires:
#   clean
#
Pr13p4:  Pr13p4.o  rectangle.o
→ g++ -o Pr13p4 Pr13p4.o rectangle.o
#
Pr13p4.o: Pr13p4.cpp rectangle.h
→ g++ Pr13p4.cpp -g -c
#
rectangle.o: rectangle.cpp rectangle.h
→ g++ rectangle.cpp -g -c
#
clean:
→ rm  -f *.o
# fin du makefile
```

ÉTAPE 7: compilation et *makefile*

Dans votre compte, localisé dans le répertoire atelier:

- Vérifier la localisation: `pwd`
- Copier le répertoire rectanglev1:
  `cp -r /grp/cours/s2gen241/Atelier/rectanglev1 .`
- Changer de répertoire: `cd rectanglev1`
- Lister : `ls -l`
- Éditer au besoin
- Compiler avec l'utilitaire make : `make`
- Lister pour voir les fichiers créés: `ls -l`
- Exécuter: `pr13p4`

ÉTAPE 8: le débogueur ddd

- Débogueur: utile pour analyser une erreur de programmation
- Compiler avec g++ en utilisant le paramètre `-g`
- Lancer le programme avec ddd:
  `ddd prog &`
- Références: guide ddd (ddd.pdf)
  `http://www.gnu.org/software/ddd/manual/`
- Référence sur l'utilisation de *ddd* qui peut être utile :
  `http://www.gnu.org/software/ddd/manual/html_mono/ddd.html#Sample%20Session`

ÉTAPE 8: utilisation de ddd

- Compiler un programme avec l'option `-g`.
- Exécuter avec ddd, par exemple: `ddd pr13p4 &`
- Placer un point d'arrêt (*breakpoint)* à un endroit stratégique (curseur en début de ligne)
- Démarrer le programme jusqu'au point d'arrêt
- Faire un *display* de quelques variables
- Exécuter pas à pas, etc.

ÉTAPE 9: archiver un projet

- Diverses commandes possibles

- En se situant au dessus du répertoire

```
zip -r archive_v1 rectanglev1
```

Conclusion

- Voilà l'essentiel pour développer des programmes simples en C++ avec g++ sous Linux-Unix.
- Il reste à poursuivre l'apprentissage de cet environnement, paradoxalement simple et complexe à la fois.