

IN301 Langage C

Introduction et environnement de programmation

Pierre Coucheney
`pierre.coucheney@uvsq.fr`

UNIVERSITÉ DE
VERSAILLES
ST-QUENTIN-EN-YVELINES



Contenus du cours

Révisions et approfondissement du C

Contenus du cours

Révisions et approfondissement du C

Environnement de programmation

Contenus du cours

Révisions et approfondissement du C

Environnement de programmation

Manipulation des fonctions de la librairie standard (entrées/sorties, chaînes de caractères...)

Contenus du cours

Révisions et approfondissement du C

Environnement de programmation

Manipulation des fonctions de la librairie standard (entrées/sorties, chaînes de caractères...)

Application à des algorithmes et structures de données vues en cours d'algo

Contenus du cours

Révisions et approfondissement du C

Environnement de programmation

Manipulation des fonctions de la librairie standard (entrées/sorties, chaînes de caractères...)

Application à des algorithmes et structures de données vues en cours d'algo

Introduction aux appels système

Références

Sur internet :

- <http://fr.openclassrooms.com/informatique/cours/apprenez-a-programmer-en-c>
(ex lesiteduzero)
- <http://c.developpez.com/cours/bernard-cassagne/hyperdoc.php>
- <http://progdupeu.pl/tutoriels/15/le-langage-c/>

“The C programming language”, second edition, Brian Kernighan et Dennis Ritchie

Existe en version française chez Dunod

Systèmes UNIX

UNIX est un système d'exploitation pour ordinateur : il fournit une interface à l'utilisateur pour contrôler les différents composants du PC.

Systèmes UNIX

UNIX est un système d'exploitation pour ordinateur : il fournit une interface à l'utilisateur pour contrôler les différents composants du PC.

Une interface en ligne de commandes est fournie qui s'appelle un **terminal**. Accessible avec CTRL+ALT+T depuis la machine virtuelle.

Systèmes UNIX

UNIX est un système d'exploitation pour ordinateur : il fournit une interface à l'utilisateur pour contrôler les différents composants du PC.

Une interface en ligne de commandes est fournie qui s'appelle un **terminal**. Accessible avec CTRL+ALT+T depuis la machine virtuelle.

On saisit les commandes à la suite du symbole \$

Exemple :

```
coucheney@chiripa~$: geany
```

exécute le programme (éditeur de texte) geany.

Systèmes UNIX

UNIX est un système d'exploitation pour ordinateur : il fournit une interface à l'utilisateur pour contrôler les différents composants du PC.

Une interface en ligne de commandes est fournie qui s'appelle un **terminal**. Accessible avec CTRL+ALT+T depuis la machine virtuelle.

On saisit les commandes à la suite du symbole \$

Exemple :

```
coucheney@chiripa~$: geany
```

exécute le programme (éditeur de texte) geany.

Pour lancer d'autres commandes durant l'exécution, on ajoute & à la fin

```
coucheney@chiripa:~$ geany &
```

Systèmes UNIX

UNIX est un système d'exploitation pour ordinateur : il fournit une interface à l'utilisateur pour contrôler les différents composants du PC.

Une interface en ligne de commandes est fournie qui s'appelle un **terminal**. Accessible avec CTRL+ALT+T depuis la machine virtuelle.

On saisit les commandes à la suite du symbole \$

Exemple :

```
coucheney@chiripa~$: geany
```

exécute le programme (éditeur de texte) geany.

Pour lancer d'autres commandes durant l'exécution, on ajoute & à la fin

```
coucheney@chiripa:~$ geany &
```

Pour stopper la commande en cours : CTRL+C

Autres fonctionnalités du terminal

Les flèches HAUT et BAS permettent de se déplacer dans l'historique des commandes.

Autres fonctionnalités du terminal

Les flèches HAUT et BAS permettent de se déplacer dans l'historique des commandes.

La commande `history` affiche l'historique des commandes du terminal.

Autres fonctionnalités du terminal

Les flèches HAUT et BAS permettent de se déplacer dans l'historique des commandes.

La commande `history` affiche l'historique des commandes du terminal.

`CTRL+R` permet de chercher la dernière commande qui contient le chaîne de caractères que l'on renseigne.

Autres fonctionnalités du terminal

Les flèches HAUT et BAS permettent de se déplacer dans l'historique des commandes.

La commande `history` affiche l'historique des commandes du terminal.

CTRL+R permet de chercher la dernière commande qui contient le chaîne de caractères que l'on renseigne.

La touche TAB permet la **complétion automatique**.

Système de fichiers

Les fichiers sont organisés selon une arborescence :

- la racine est le répertoire /
- les fichiers sont les feuilles de l'arbre, sinon ce sont des répertoires.

Système de fichiers

Les fichiers sont organisés selon une arborescence :

- la racine est le répertoire /
- les fichiers sont les feuilles de l'arbre, sinon ce sont des répertoires.

Quand on ouvre un terminal, on est par défaut dans le **répertoire personnel**

```
/home/moi
```

Système de fichiers

Les fichiers sont organisés selon une arborescence :

- la racine est le répertoire /
- les fichiers sont les feuilles de l'arbre, sinon ce sont des répertoires.

Quand on ouvre un terminal, on est par défaut dans le **répertoire personnel**

```
/home/moi
```

Pour se déplacer dans l'arbre : commande `cd` (change directory).

Système de fichiers

Les fichiers sont organisés selon une arborescence :

- la racine est le répertoire /
- les fichiers sont les feuilles de l'arbre, sinon ce sont des répertoires.

Quand on ouvre un terminal, on est par défaut dans le **répertoire personnel**

```
/home/moi
```

Pour se déplacer dans l'arbre : commande `cd` (change directory).

Pour connaître où l'on est : commande `pwd` (print working directory).

Système de fichiers

Les fichiers sont organisés selon une arborescence :

- la racine est le répertoire /
- les fichiers sont les feuilles de l'arbre, sinon ce sont des répertoires.

Quand on ouvre un terminal, on est par défaut dans le **répertoire personnel**

```
/home/moi
```

Pour se déplacer dans l'arbre : commande `cd` (change directory).

Pour connaître où l'on est : commande `pwd` (print working directory).

Deux possibilités pour désigner un répertoire/fichier : **chemin absolu** ou **chemin relatif**.

Répertoire parent : “ . . ”

Les commandes de base

- Aide sur une commande : `man commande`

Les commandes de base

- Aide sur une commande : `man commande`
- Lister le contenu d'un répertoire : `ls [option] [chemin]`

Les commandes de base

- Aide sur une commande : `man commande`
- Lister le contenu d'un répertoire : `ls [option] [chemin]`
- Construire un répertoire : `mkdir [option] chemin`

Les commandes de base

- Aide sur une commande : `man commande`
- Lister le contenu d'un répertoire : `ls [option] [chemin]`
- Construire un répertoire : `mkdir [option] chemin`
- Copier un fichier : `cp source destination`

Les commandes de base

- Aide sur une commande : `man commande`
- Lister le contenu d'un répertoire : `ls [option] [chemin]`
- Construire un répertoire : `mkdir [option] chemin`
- Copier un fichier : `cp source destination`
- Déplacer un fichier : `mv source destination`

Les commandes de base

- Aide sur une commande : `man commande`
- Lister le contenu d'un répertoire : `ls [option] [chemin]`
- Construire un répertoire : `mkdir [option] chemin`
- Copier un fichier : `cp source destination`
- Déplacer un fichier : `mv source destination`
- Supprimer (définitivement !) un fichier : `rm fichier`

Les commandes de base

- Aide sur une commande : `man commande`
- Lister le contenu d'un répertoire : `ls [option] [chemin]`
- Construire un répertoire : `mkdir [option] chemin`
- Copier un fichier : `cp source destination`
- Déplacer un fichier : `mv source destination`
- Supprimer (définitivement !) un fichier : `rm fichier`
- Afficher le contenu d'un fichier texte : `cat` ou `less`

Les commandes de base

- Aide sur une commande : `man commande`
- Lister le contenu d'un répertoire : `ls [option] [chemin]`
- Construire un répertoire : `mkdir [option] chemin`
- Copier un fichier : `cp source destination`
- Déplacer un fichier : `mv source destination`
- Supprimer (définitivement !) un fichier : `rm fichier`
- Afficher le contenu d'un fichier texte : `cat` ou `less`
- Afficher le nombre de lignes/mots/caractères d'un fichier : `wc -l` ou `wc -w` ou `wc -c`

Compilation d'un programme en C

Un programme `toto.c` peut être compilé à partir du terminal avec la commande `gcc`

```
gcc toto.c
```

Cela produit un exécutable qui se nomme par défaut `a.out`

La commande pour lancer l'exécutable depuis le terminal est

```
./a.out
```

Compilation d'un programme en C

Un programme `toto.c` peut être compilé à partir du terminal avec la commande `gcc`

```
gcc toto.c
```

Cela produit un exécutable qui se nomme par défaut `a.out`

La commande pour lancer l'exécutable depuis le terminal est

```
./a.out
```

Les principales options de compilation :

- `gcc -o nom_du_prog toto.c` permet de nommer l'exécutable `nom_du_prog`
- l'option `-Wall` affiche les warnings de compilation,
- l'option `-lm` permet d'utiliser la librairie mathématiques.

En général on utilisera la commande

```
gcc -Wall -lm -o nom toto.c
```

Redirection des entrées et sorties

Les commandes tapées dans le terminal affichent leur résultat sur la **sortie standard** qui est le terminal.

Redirection des entrées et sorties

Les commandes tapées dans le terminal affichent leur résultat sur la **sortie standard** qui est le terminal.

L'opérateur `>` permet de rediriger la sortie standard d'un programme, par exemple vers un fichier.

Exemple :

```
ls > titi
```

écrira le résultat de la commande `ls` dans le fichier `titi`.

Redirection des entrées et sorties

Les commandes tapées dans le terminal affichent leur résultat sur la **sortie standard** qui est le terminal.

L'opérateur > permet de rediriger la sortie standard d'un programme, par exemple vers un fichier.

Exemple :

```
ls > titi
```

écrira le résultat de la commande `ls` dans le fichier `titi`.

L'opérateur < permet de rediriger l'entrée standard.

Redirection des entrées et sorties

Les commandes tapées dans le terminal affichent leur résultat sur la **sortie standard** qui est le terminal.

L'opérateur > permet de rediriger la sortie standard d'un programme, par exemple vers un fichier.

Exemple :

```
ls > titi
```

écrira le résultat de la commande `ls` dans le fichier `titi`.

L'opérateur < permet de rediriger l'entrée standard.

L'opérateur | permet de rediriger la sortie standard d'un programme vers l'entrée standard d'un autre programme (tube).

Exemple :

```
ls | wc -w
```

```
ping -c 5 www.google.com|head -n 6|cut -f8,8 -d ' '|sort
```

Un nouvel outil : le contrôle de version

Sauvegarder ne suffit pas :

- seule la dernière version est conservée ;
- on perd tout si l'ordinateur est cassé / volé... il faut faire des copies distantes.

Un nouvel outil : le contrôle de version

Sauvegarder ne suffit pas :

- seule la dernière version est conservée ;
- on perd tout si l'ordinateur est cassé / volé... il faut faire des copies distantes.

Le logiciel dropbox permet de réaliser des copies incrémentales sur un répertoire distant.

Un nouvel outil : le contrôle de version

Sauvegarder ne suffit pas :

- seule la dernière version est conservée ;
- on perd tout si l'ordinateur est cassé / volé... il faut faire des copies distantes.

Le logiciel dropbox permet de réaliser des copies incrémentales sur un répertoire distant.

Comment faire pour travailler à plusieurs sur un fichier ?

Un nouvel outil : le contrôle de version

Sauvegarder ne suffit pas :

- seule la dernière version est conservée ;
- on perd tout si l'ordinateur est cassé / volé... il faut faire des copies distantes.

Le logiciel dropbox permet de réaliser des copies incrémentales sur un répertoire distant.

Comment faire pour travailler à plusieurs sur un fichier ?

dropbox ne permet pas de résoudre les conflits.

On utilisera le logiciel git.

Git : les commandes de base

Créer un répertoire qui sera sous contrôle de version :

```
git init nouveau_projet
```


Git : les commandes de base

Créer un répertoire qui sera sous contrôle de version :

```
git init nouveau_projet
```

Ajouter les fichiers à sauvegarder :

```
git add toto.c titi.c
```

Git : les commandes de base

Créer un répertoire qui sera sous contrôle de version :

```
git init nouveau_projet
```

Ajouter les fichiers à sauvegarder :

```
git add toto.c titi.c
```

Valider des modifications ou des ajouts :

```
git commit -a
```

Git : les commandes de base

Créer un répertoire qui sera sous contrôle de version :

```
git init nouveau_projet
```

Ajouter les fichiers à sauvegarder :

```
git add toto.c titi.c
```

Valider des modifications ou des ajouts :

```
git commit -a
```

Examiner l'état du répertoire

```
git status
```

Git : les commandes de base

Créer un répertoire qui sera sous contrôle de version :

```
git init nouveau_projet
```

Ajouter les fichiers à sauvegarder :

```
git add toto.c titi.c
```

Valider des modifications ou des ajouts :

```
git commit -a
```

Examiner l'état du répertoire

```
git status
```

Voir l'historique des modifications

```
git log
```

Git : les commandes de base

Créer un répertoire qui sera sous contrôle de version :

```
git init nouveau_projet
```

Ajouter les fichiers à sauvegarder :

```
git add toto.c titi.c
```

Valider des modifications ou des ajouts :

```
git commit -a
```

Examiner l'état du répertoire

```
git status
```

Voir l'historique des modifications

```
git log
```

Récupérer une ancienne version

```
git checkout numero_version
```

Travailler avec un dépôt distant

En TD, on utilisera l'hébergeur de projets `github`.

Travailler avec un dépôt distant

En TD, on utilisera l'hébergeur de projets `github`.

Une fois votre compte validé, vous créerez un répertoire IN301 qui contiendra votre travail du semestre (TD + projet).

Travailler avec un dépôt distant

En TD, on utilisera l'hébergeur de projets github.

Une fois votre compte validé, vous créerez un répertoire IN301 qui contiendra votre travail du semestre (TD + projet).

Pour copier ce répertoire sur votre machine :

```
git clone https://github.com/moi/IN301
```

où moi doit être remplacé par votre nom d'utilisateur.

Travailler avec un dépôt distant

En TD, on utilisera l'hébergeur de projets github.

Une fois votre compte validé, vous créerez un répertoire IN301 qui contiendra votre travail du semestre (TD + projet).

Pour copier ce répertoire sur votre machine :

```
git clone https://github.com/moi/IN301
```

où moi doit être remplacé par votre nom d'utilisateur.

Vous travaillez sur la version locale comme avant.

Travailler avec un dépôt distant

En TD, on utilisera l'hébergeur de projets `github`.

Une fois votre compte validé, vous créerez un répertoire `IN301` qui contiendra votre travail du semestre (TD + projet).

Pour copier ce répertoire sur votre machine :

```
git clone https://github.com/moi/IN301
```

où `moi` doit être remplacé par votre nom d'utilisateur.

Vous travaillez sur la version locale comme avant.

Avant de finir, vous mettez à jour le répertoire distant (sur `github`)

```
git push
```

Bonne programmation : Objectifs

Bonne programmation : Objectifs

- efficacité
- fiabilité
- robustesse, sécurité
- maintenance du code, développement à plusieurs
- portabilité
- vitesse de développement

Structuration d'un programme :

- découpage en fichiers et fonctions cohérents
- chaque brique (instruction, fonction, fichier) doit être relativement courte

Lisibilité :

Lisibilité :

- usage des commentaires
- choix d'identificateurs explicites
- instructions courtes et simples
- indentation adaptée
- éviter les commandes inhabituelles, telles GOTO LBL BREAK CONTINUE et les variables globales