Algorithmique et Programmation, IMA

Cours 1 : Introduction et Hello World

Laure Gonnord
http://laure.gonnord.org/pro/teaching/
Laure.Gonnord@polytech-lille.fr

Université Lille 1 - Polytech Lille

Systèmes informatiques

Un système informatique :

- est conçu pour automatiser le traitement d'une tâche
- est divisé en matériel (stockage, periphériques, unité centrale, . . .) et logiciel.
- logiciels/applications : gestion, jeux, bureautique, traitement de données, . . .
- un système d'exploitation fait le lien et gère les ressources

Systèmes informatiques

Différents types :

- temps réel (contraintes de temps) : contrôle d'une chaîne de production, ou le contrôle commande d'un avion.
- Systèmes transactionnels : contrôles de bases de données, ...
- embarqués (pda), distribués (réservation de train), ...

Développement logiciel

Création, développement de logiciels suivant les besoins et les offres matérielles :

- Systèmes complexes
- Gros logiciels

Programmation structurée

Objectifs de l'enseignement :

- **Conception** de bout en bout d'un logiciel : du cahier des charges à l'implémentation et la documentation.
- **Analyse** du problème initial et hiérarchisation des priorités : notion de sous problème.
- Réflexions sur la pertinence des solutions et leur coût d'exécution : **Analyses de complexité**
- Travail contraint en équipe et en temps : **Réalisation de projet**.

Placement dans les enseignements IMA

Cours prérequis des enseignements de : réseaux, systèmes, programmation avancée, programmation objet, ...

Supports

- 1 polycopié référence de C.
- Transparents de cours à avoir sur soi en TP.
- Énoncés de td et de tp.

Enseignement différentié

Mise en œuvre :

- Une voire « normale » : 6 cours 7 td 12 tp
- Renforcement maths : dispense des cours **Penser à récupérer les supports papier**
- Renforcement info : 20h supplémentaires (EDi dans l'emploi du temps).

Environnement Linux

Environnement de tp :

- Ordinateurs en réseau avec compte NFS (non local)
- Les TPs sont réalisés sous Linux avec un environnement graphique « Konquéror ».
- Les outils libres sont privilégiés.

Premier programme

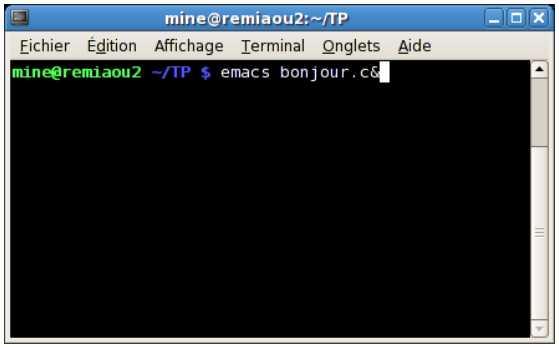
```c
#include <stdio.h>

int main()
{
    printf("Bonjour tout le monde!\n");
    return 0;
}
```

Effet :

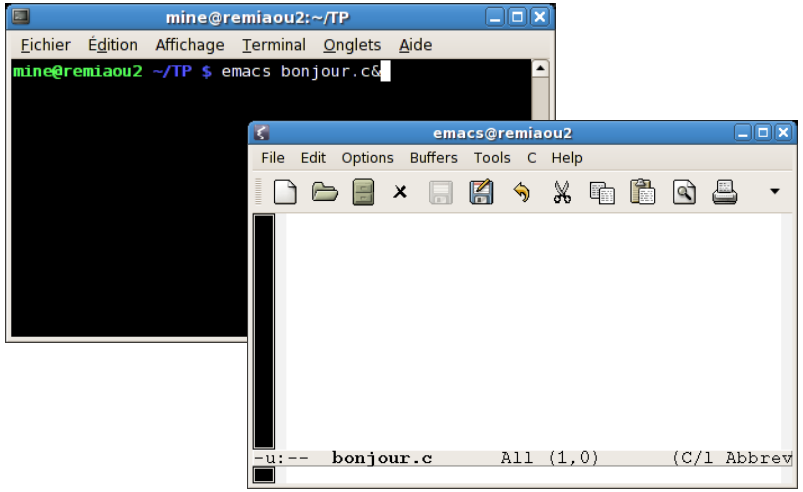
- affiche `Bonjour tout le monde !`,
- retourne le code 0 (tout s'est bien passé).

Compilation et exécution



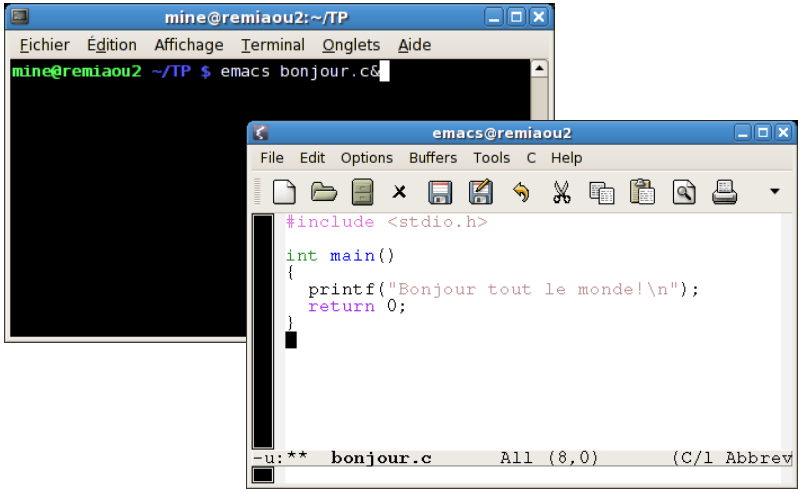
Lancement de l'éditeur en tâche de fond (&).

Compilation et exécution



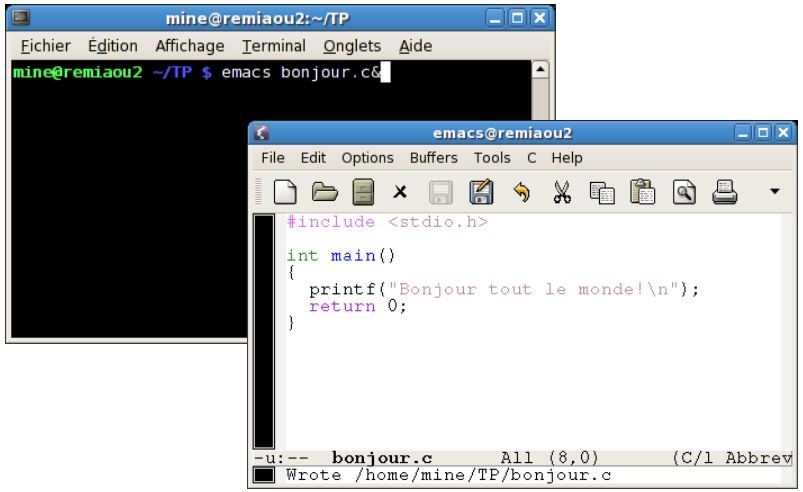
Lancement de l'éditeur en tâche de fond (&).

Compilation et exécution



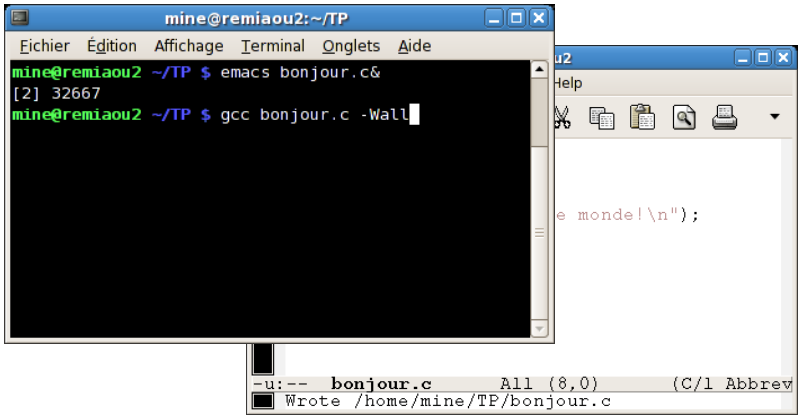
On tape le texte du programme.

Compilation et exécution



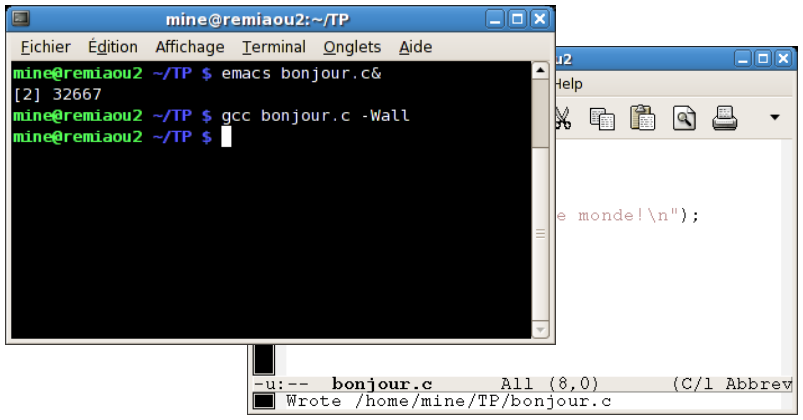
Il ne faut pas oublier de sauvegarder.

Compilation et exécution



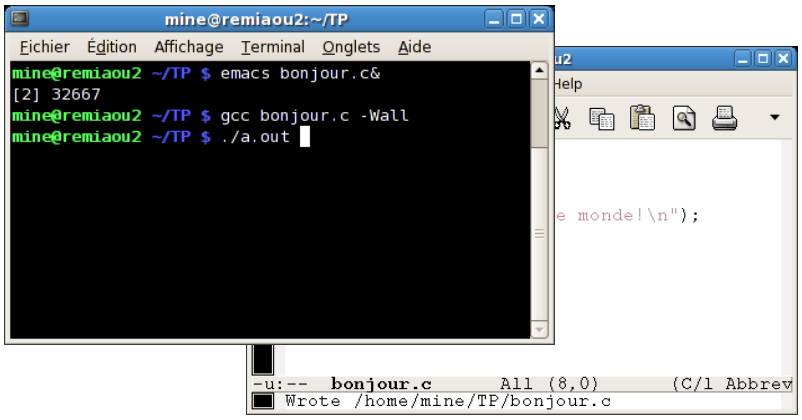
Lancement de la compilation avec `gcc`.

Compilation et exécution



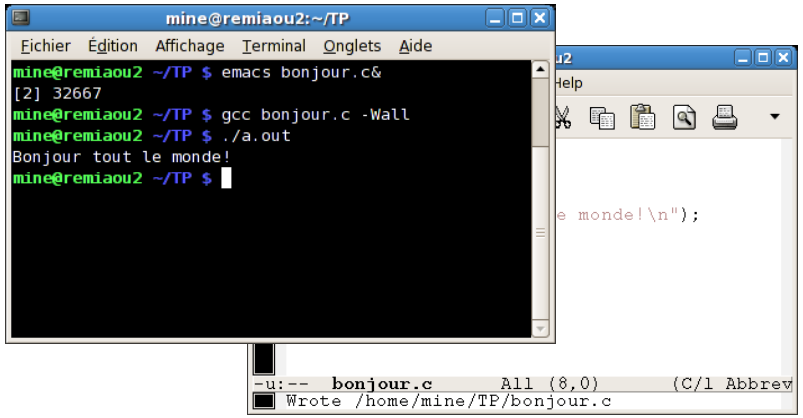
Si le compilateur ne dit rien, tout s'est bien passé.
Un fichier `a.out` a été créé.

Compilation et exécution



Lancement de l'exécutable.

Compilation et exécution



Le programme s'exécute et rend la main.

Anatomie de `bonjour.c`

```
#include <stdio.h>

int main()
{
    printf("Bonjour tout le monde!\n");
    return 0 ;
}
```

Tout programme C doit contenir une fonction appelée `main`.
L'exécution commence au début de `main`.

Anatomie de `bonjour.c`

```
#include <stdio.h>

int main()
{
    printf("Bonjour tout le monde!\n");
    return 0;
}
```

Par convention, la fonction `main` renvoie un code de retour :

- il est de type `int` (entier),
- la convention est de retourner 0 si tout se passe bien,
- les parenthèses de `return` sont facultatives,
- le code de retour est exploitable depuis le shell.

Anatomie de `bonjour.c`

```
#include <stdio.h>

int main()
{
    printf("Bonjour tout le monde!\n");
    return 0 ;
}
```

La fonction `printf` permet d'écrire sur l'écran.

- elle fait partie de la bibliothèque C standard,
- elle doit être importée depuis l'en-tête `stdio.h`.

Anatomie de `bonjour.c`

```c
#include <stdio.h>

int main()
{
    printf("Bonjour tout le monde!\n");
    return(0);
}
```

`printf` prend en argument une chaîne de caractères :

- tapée entre guillemets ",
- \ sert à entrer des caractères spéciaux :
  \n signifie "retour à la ligne".

Exemple d'erreur

```c
1 #include <stdio.h>
2
3 int main()
4 {
5     printf("Bonjour tout le monde!\n")
6     return(0);
7 }
```


Exemple d'erreur

```c
1 #include <stdio.h>
2
3 int main()
4 {
5     printf("Bonjour tout le monde!\n")
6     return(0);
7 }
```

Commande :

```
$ gcc bonjour.c -Wall
```

Résultat :

```
bonjour.c: In function 'main':
bonjour.c:6: error: expected ';' before 'return'
bonjour.c:7: warning : control reaches end of ...
```

▶ Il manque un `;`. Aucun `a.out` n'est généré.

Exemple d'avertissement

```c
1 int main()
2 {
3     printf("Bonjour tout le monde!\n");
4     return(0);
5 }
```

```
$ gcc bonjour.c -Wall

bonjour.c: In function 'main':
bonjour.c:3: warning: implicit declaration of function
'printf'
```

▶ Il manque un `#include <stdio.h>`.
C'est un avertissement non fatal généré par `-Wall`.

Les options `-Wall` et `-Wextra`

`-Wall` attire l'attention, entres autres, sur :

- les oublis d'imports `#include`,
- les ambiguïtés syntaxiques courantes,
- les incohérences de types.

La norme est très laxiste ne considère pas ces points comme des erreurs !

`-Wextra` ajoute des avertissements supplémentaires.

▶ Toujours compiler avec `-Wall` au moins.

Espacement

L'espacement et les sauts de lignes sont libres.

```
      # include <stdio.h>
int  main                          (
       ){
printf
  ("toto\n"
 );return(0)    ;}
```

Exceptions :

- `#include <stdio.h>` doit être sur une seule ligne,
- les sauts de ligne comptent dans les chaînes de caractères.

Commentaires

Commentaires : tout ce qui est entre `/*` et `*/` est ignoré.

```
#include <stdio.h> /* pour avoir printf */

/* la fonction principale
*/
int main(/* rien ici */)
{
   printf("toto\n");
   return(0); /* OK */
}
```

Conseils : - indentez votre code (TAB sous Emacs),
- commentez votre code.