Haute École Léonard de Vinci
SCIENCES ET TECHNIQUES

I2011 Langage C : bases

Introduction

Anthony Legrand
Jérôme Plumat

Premier programme


```
#include <stdlib.h>
#include <stdio.h>

int main() {
    printf("Hello world\n");
    exit(0);
}
```

Premier programme


```
#include <stdlib.h>
#include <stdio.h>

int main() {
    printf("Hello world\n");
    exit(0);
}
```

directives du préprocesseur (préfixées par #)

→ inclusion du contenu des fichiers *stdlib.h* et *stdio.h* dans notre code source avant la compilation

Premier programme


```
#include <stdlib.h>
#include <stdio.h>

int main() {
    printf("Hello world\n");
    exit(0);
}
```

ligne vide, ignorée par le compilateur

Premier programme


```
#include <stdlib.h>
#include <stdio.h>

int main() {
    printf("Hello world\n");
    exit(0);
}
```

main = nom de la fonction principale

→ point d'entrée de toute application C

Premier programme


```
#include <stdlib.h>
#include <stdio.h>

int main() {
    printf("Hello world\n");
    exit(0);
}
```

appel de la fonction *printf()*

→ affiche la chaîne de caractères « Hello world » sur la sortie standard (écran)

Premier programme


```
#include <stdlib.h>

#include <stdio.h>


int main() {

    printf("Hello world\n");

    exit(0);

}
```

instruction permettant de quitter l'application en renvoyant l'*exit code* ou *status code* 0 (⇔ fin normale du processus) au shell appelant

→ Affichage sous Linux avec la commande: `echo $?`

Compilation

Compilation en ligne de commande *bash*:

```
cc pgm1.c
```

→ génération d'un programme exécutable, appelé par défaut *a.out*, dans le répertoire courant

```
cc -o pgm1 pgm1.c
```

→ l'option -o permet de spécifier un autre nom pour notre exécutable

→ pas d'extension aux exécutables en Linux

Exécution

Exécution du programme dans un terminal :

```
./a.out
```

→ Si le nom par défaut a été conservé à la compilation

```
./pgm1
```

→ Si le nom `pgm1` a été défini à la compilation avec l'option -o

Compilateur

- **cc** (*C Compiler*) désigne le compilateur C par défaut des systèmes Unix/Linux
- Sous Linux et en particulier la distribution Ubuntu, cc est un lien symbolique vers **gcc** (*GNU C Compiler*), le compilateur C de la suite de logiciels libres de compilation GCC (*GNU Compiler Collection*, fournissant notamment G++ pour C++, GobjC pour Objective-C, GCJ pour Java)

Options de compilation

- Limitation de l'utilisation de *gcc* via des **options de compilation** (cf. commande *alias*):

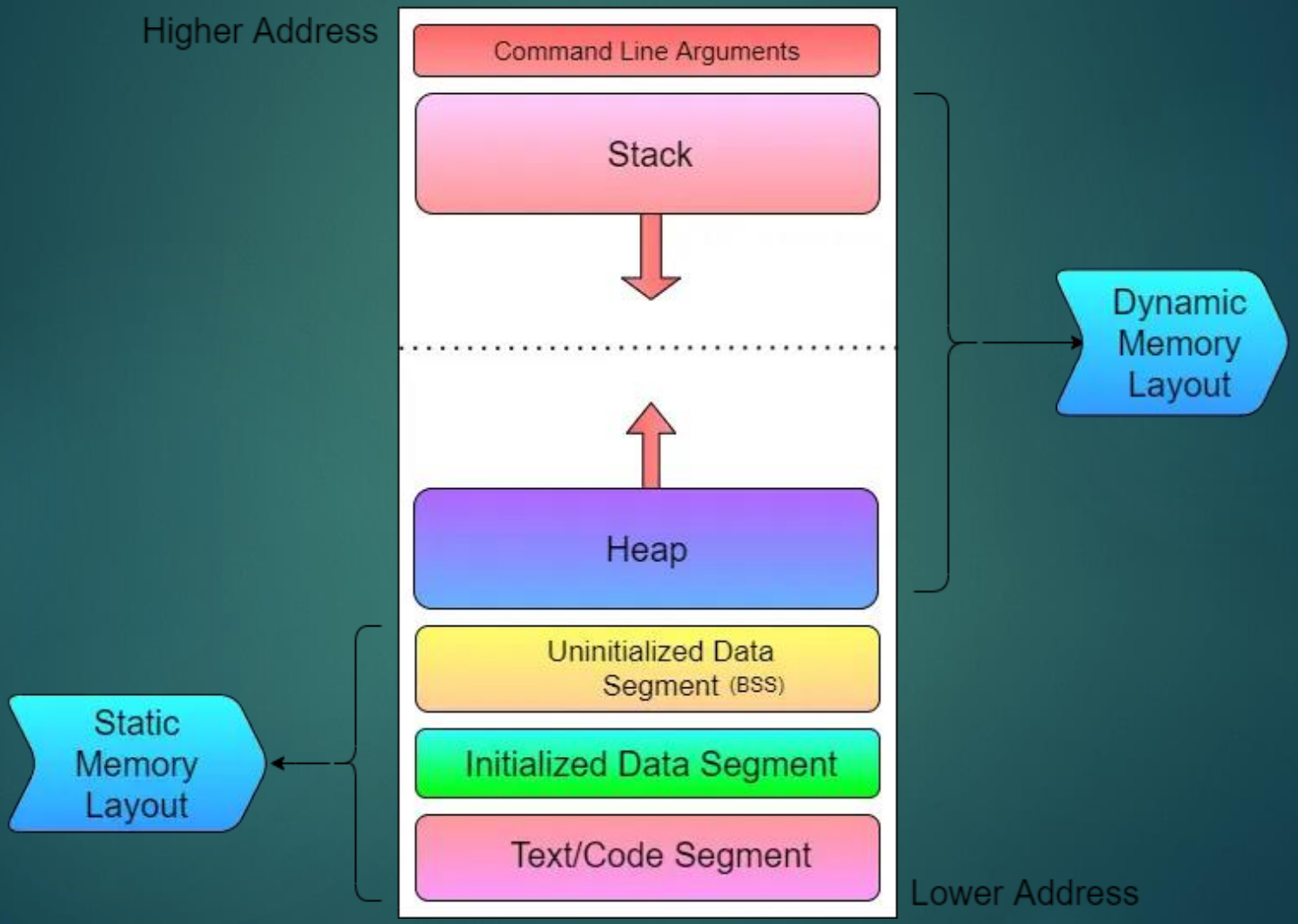
```
gcc -std=c11 -pedantic
    -Werror -Wall -Wvla -Wno-unused-variable
    -o pgm1 pgm1.c
```

Enregistrement des options de compilation

- Pour vous éviter de taper les options à chaque compilation, la ligne suivante a été ajoutée à votre fichier `~/.bashrc` , script de configuration du bash (`~/.bash_profile` pour MacOSX):

```
alias cc='gcc -std=c11 -pedantic -Werror -Wall -Wvla
      -Wno-unused-variable'
```

Rappel: organisation de la mémoire virtuelle d'un processus



Source: https://hackthedeveloper.com/memory-layout-c-program/

Rappel: organisation de la mémoire virtuelle d'un processus

- **Text/Code Segment** : contains the executable binary, the instructions to be followed by the processor.
- **Data Segment** : split in two parts. One part contains all the initialized data variables (such as global variable or static variables). The other part enfolds the uninitialized data of the program, which includes the global variables which have not been initialized.
- **Heap** : the memory segment used for all the dynamic memory allocations (`malloc()`, `calloc()`...).
- **Stack**: contains all the local variables.
- **Command-line arguments** : arguments of shell command (*argv*) and environment variables