

TP4 – make et pointeurs

Langage C (LC4)

semaine du 17 février

1 Premiers pas avec make

1.1 L'outil make sans Makefile

Considérez un ou plusieurs programmes des TP précédents. Vérifiez le sens de l'option `-c` de `gcc`, puis

- observez le temps d'exécution de la commande `gcc -c <votre programme>.c` (commande `time`),
- puis observez le temps d'exécution de la commande `make <votre programme>.o`.
- Effacez le fichier `<votre programme>.o` de votre répertoire courant et recommencez l'opération ci-dessus.

Question 1. Qu'en concluez vous à propos de l'utilisation de `make` ?

1.2 Compilation séparée et premier Makefile

Considérez les trois programmes suivants, et copiez les dans trois fichiers distincts que vous nommerez respectivement : `hello.c`, `hello.h`, `main.c`.

```
#include <stdio.h>

void hello()
{
    printf("Hello World\n");
}
```

```
#ifndef H_HELLO
#define H_HELLO

void hello();

#endif
```

```
#include <stdio.h>
#include "hello.h"

int main()
```

```
{  
    hello();  
    return 0;  
}
```

Question 2. Une fois compilé que produit l'exécution de ce programme ?

Question 3. À quoi semble servir la suite de directives **#ifndef**, **#define**, et **#endif** ?

Question 4. Établissez un ordre de dépendance entre les fichiers ci-dessus et les fichiers intermédiaires que vous pourriez construire au cours de la compilation (les fichiers objets, l'exécutable).

Question 5. En suivant le tutoriel <http://gl.developpez.com/tutoriel/outil/makefile/>, créez un Makefile simple pour compiler le programme ci-dessus, tout en générant tous les fichiers objets intermédiaires.

Question 6. Améliorez votre Makefile afin qu'il vous permette aussi de nettoyer votre répertoire, c'est à dire d'en retirer tous les fichiers objets. Il s'agit d'écrire une règle supplémentaire dont la cible sera conventionnellement appelée **clean**.

Question 7. Ajoutez une autre règle qui permette de retirer tous les fichiers générés au cours de la compilation. Cela vous permettra de reconstruire complètement vos programmes.

Dans les prochains exercices et TPs, il est désormais recommandé de créer un dossier spécifique et un Makefile spécifique à ce dossier. Pour ne pas aller trop vite nous vous fournissons un schéma de Makefile que vous pouvez modifier pour compiler vos exercices.

Normalement, il suffit d'insérer les noms de vos fichiers ***.c** dans la première ligne, et ensuite vous pouvez compiler votre programme avec la commande **make** dans le terminal.

```
SOURCES=fichier1.c fichier2.c ...  
OBJECTS=$(SOURCES:.c=.o)  
CFLAGS= -std=c89 -Wall -Wextra  
EXECUTABLE=exo  
  
$(EXECUTABLE): $(OBJECTS)  
gcc -o $(EXECUTABLE) $(OBJECTS)  
  
%.o : %.c  
gcc -c $(CFLAGS) $< -o $@  
  
clean:  
    rm -f $(EXECUTABLE) $(OBJECTS)
```

Question 8. Avant de passer à la suite essayez de comprendre le sens du Makefile proposé.

2 Les pointeurs

Question 9. Recopiez, compilez et exécutez le programme `ptr-return.c` suivant :

```
#include <stdio.h>

int *toto()
{
    int x = 42;
    return &x;
}

void tata()
{
    int a = 23;
}

int main(int argc, char *argv[])
{
    int *y = toto();
    tata();
    printf("%d\n", *y);
    return(0);
}
```

Essayez de comprendre ce qui s'est passé.

Question 10. Écrire une fonction `char *recherche(char *s, char c)` qui renvoie un pointeur vers la première occurrence dans la chaîne `s` du caractère `c` passé en argument. Si ce caractère n'apparaît pas dans la chaîne, la fonction devra renvoyer `NULL`.

Question 11. À l'aide de la fonction précédente, écrire une fonction `int compte(char *s, char c)` qui renvoie le nombre d'occurrences de `c` dans `s`.