Créons maintenant un nouveau fichier appelé Makefile contenant,

```
printString: main.o printString.o
    gcc -o printString main.o printString.o

main.o:
    gcc -c main.c

printString.o:
    gcc -c printString.c

clean:
    rm -f *.o
```

Cette fois, taper make va provoquer la compilation de chaque fichier .c pour produire des fichiers .o. Ces derniers seront ensuite liés ensemble pour former un exécutable appelé printString. La cible printString est la première cible du fichier et par conséquent celle par défaut. Quand make est lancé, il vérifie les dépendances de printString, c'est-à-dire que les fichiers main.o et printString.o existent et ne sont pas plus récents que la cible printString. Si l'un ou l'autre des fichiers n'existe pas, alors la cible pour le construire est exécutée. Toute cible autre que celle par défaut peut être lancée en tapant son nom après la commande make par exemple make clean

Écrire des fichiers make make dans lesquels chacun des noms de fichier doit être spécifié peut rapidement devenir chronophage. Au lieu de spécifier des cibles de manière explicite, il est possible d'utiliser des variables automatiques,

```
printIt: main.o printString.o
    gcc -o $@ $^

%.o: %.c
    gcc -c $< -o $@

clean:
    rm -f *.o
```

Ce Makefile a exactement le même comportement que le précédent. La variable automatique $@ correspond au nom de la cible, $^ sont les noms de toutes les dépendances, et $< est le nom du premier prérequis. Pour chaque fichier .o nécessaire à la cible par défaut, make essaiera le joker %.c. Si le fichier .c est manquant, make signalera une erreur.

Des jokers peuvent aussi être utilisés pour définir une liste d'objets à partir des fichiers .c présents dans le répertoire courant,

```
OBJECTS = $(patsubst %.c,%.o, $(wildcard *.c))

printIt: $(OBJECTS)
    gcc -o $@ $^

%.o: %.c
    gcc -c $< -o $@
```

où OBJECTS est une variable. Dans ce cas, tous les fichiers .c du répertoire en cours d'utilisation seront utilisés pour construire un exécutable appelé printIt. L'instruction wildcard énumère tous les fichiers qui correspondent au motif *.c. Ensuite, patsubst enlève le .c final et le remplace par .o. La liste résultante est affectée à la variable OBJECTS Essayez d'utiliser la commande touch pour actualiser l'horodatage de chaque fichier et relancez make pour voir ce qui se passe.

Les fichiers Make peuvent avoir plusieurs niveaux de dépendances. Pour la distribution de logiciels sur plusieurs plateformes, les Makefiles sont souvent générés à partir de modèles grâce à l'outil autoconf

***Article de W. H. Bell***