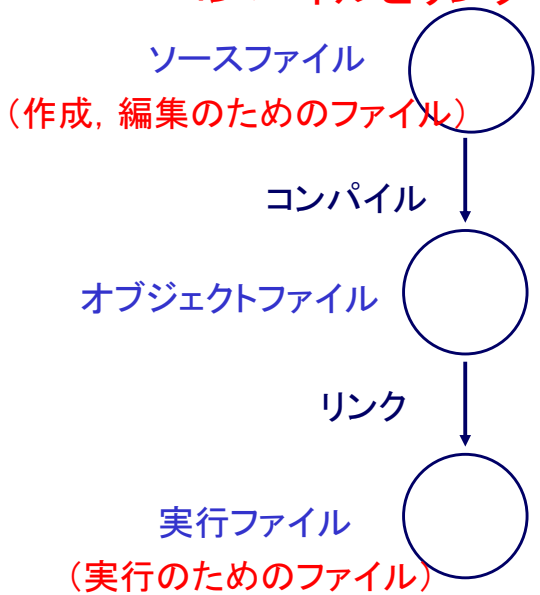
make の使い方

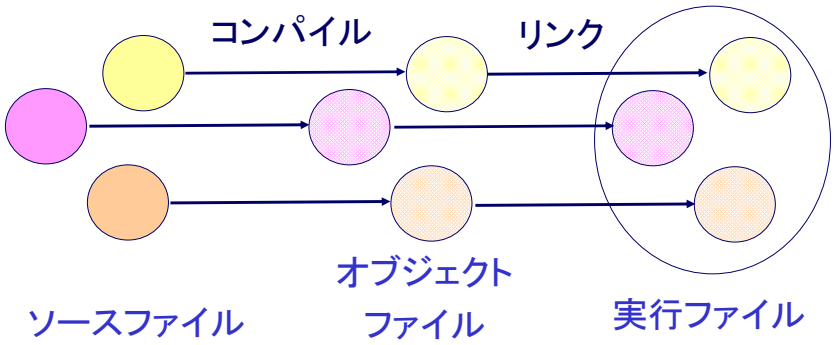
ファイルの種類

- ソースファイル
  - プログラマーが作成するプログラムファイルのこと
- オブジェクトファイル
  - C＋＋ソースファイルごとに自動的に作られるファイル
- 実行ファイル
  - 実行可能なプログラムファイル

コンパイルとリンク



分割コンパイル



- 個別のソースファイル別にコンパイルを行うこと

ファイルの依存関係

•例えばprgA.c , prgB.c , prgC.c からプログラムを作るとする

```
$ cc -c prgA.c
$ cc -c prgB.c
$ cc -c prgC.c
$ cc prgA.o prgB.o prgC.o -o prg
```

コンパイルコマンドは上記の通り

•ファイルの依存関係は次のようになっている

prg: prgA.o prgB.o prgC.oから作られる
prgA.o: prgA.cのみから作られる
prgB.o: prgB.cのみから作られる
prgC.o: prgC.cのみから作られる

make とは

- 使用法
  - ファイルの依存関係そのものを，あらかじめファイルとして記述しておく（最初に1度だけ）
- make の効用
  - make とタイプするだけで，複数のコンパイルコマンド が自動実行される
  - バッチファイルとの違い： 必要なソースファイルだけを再コンパイルする機能を有する

•ファイルの依存関係をMakefileに「ターゲット：ソース」の形で書く

```
CC= cc
CFLAGS= -O

prg: prgA.o prgB.o prgC.o
prgA.o: prgA.c
prgB.o: prgB.c
prgC.o: prgC.c
```

•最初のCC=はコンパイラを指定し、CFLAGS=はコンパイルオプションを指定している。（マクロの項参照）

•makeを実行すると次のようになる

```
$ make
cc -O -c prgA.c -o prgA.o
cc -O -c prgB.c -o prgB.o
cc -O -c prgC.c -o prgC.o
cc prgA.o prgB.o prgC.o -o prg
```

Makefile の例
― UNIX の場合 ―

```
CC=cc
CFLASG=-O
main: main.o count.o
    cc –o main main.o count.o
main.o : main.c
    cc –c –o main.o main.c
count.o : count.c
    cc –c –o count.o count.c
```

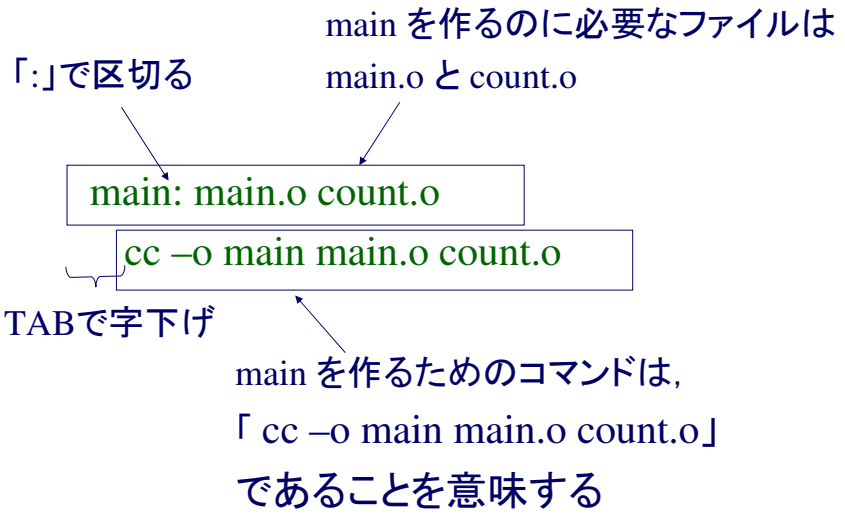
拡張子
ソースファイル： .c
オブジェクトファイル：.o

Makefile の例
― Windows の場合 ―

```
CC=cc
CFLASG=-O
main: main.obj count.obj
    cc –o main main.obj count.obj
main.obj : main.cpp
    cc –c –o main.obj main.cpp
count.obj : count.cpp
    cc –c –o count.obj count.cpp
```

拡張子
ソースファイル： .cpp
オブジェクトファイル： .obj

Makefile の書き方



makeについて

- makeというは、Makefile に記述された「ファイルの依存関係」に従って，各種コマンドの自動実行を行なうユーティリティ
- Makefileはテキストファイルであり、通常プログラムのソースファイルと同じディレクトリに置く
- make の使用法

  $ make

  とコマンドを入力することでコンパイルする
- makeを用いるとファイルの依存関係を毎回指定することなしにコンパイルできる。また、更新したファイルだけコンパイルするので、コンパイルの時間を短縮できる

マクロ

•先ほどの例では、CC＝ccと書いて、利用するコンパイラを指定した。これは makeがCコンパイラ名として、CCという変数(マクロ)に代入された文字列 を利用するという性質を利用している
（CFLAGSも同様）

•これ以外にもマクロをユーザが定義して使うことが可能

```
OBJS=argA.o argB.o argC.o
arg: $(OBJS)
```

⟺ 同じ意味

```
Arg: argA.o argB.o argC.o
```

•マクロを利用すれば同じ事を何度も書く必要がなくなる