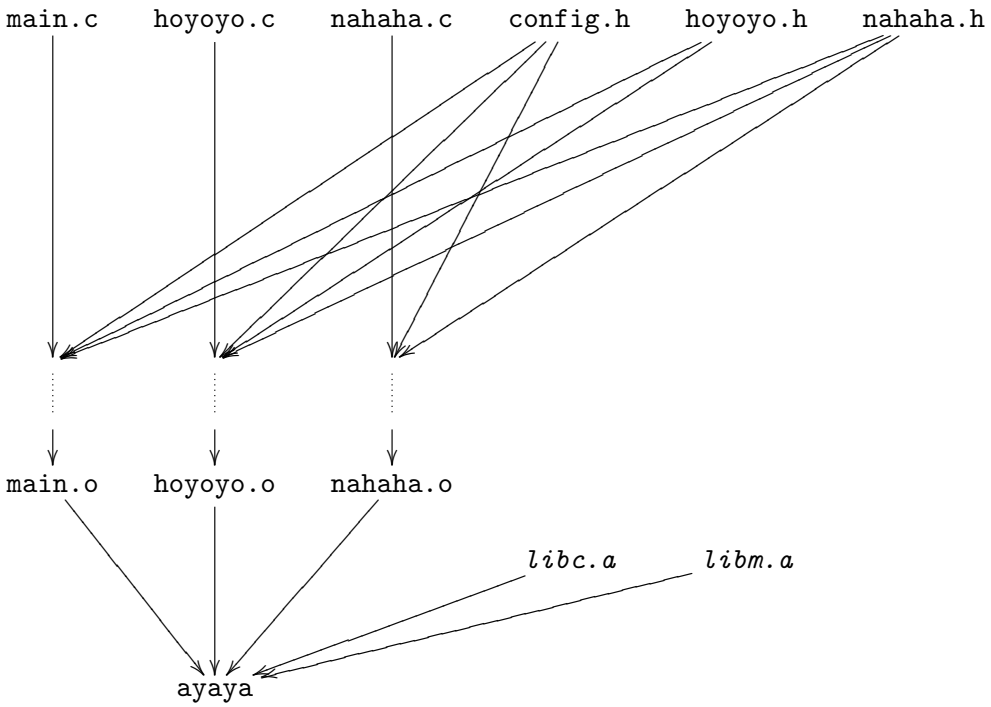
1 初歩

1.1 make が必要な理由

```
cc -o ayaya main.c hoyoyo.c nahaha.c -lm
```

を実行すると，たとえば、次のような処理が行われる。



ここで、hoyoyo.c だけを修正したとき、全部を再実行するのは時間の無駄である。
途中の生成物を残すようにするには，

```
cc -c main.c
cc -c hoyoyo.c
cc -c nahaha.c
cc -o ayaya main.o hoyoyo.o nahaha.o -lm
```

を実行する。hoyoyo.c だけを修正したときは、再コンパイルには

```
cc -c hoyoyo.c
cc -o ayaya main.o hoyoyo.o nahaha.o -lm
```

だけを実行すると時間が節約できる。しかし、それを人間が毎回判断することは現実的ではない。

- 手動で行うと間違いやすい。というより、ある程度以上の規模だと手動だと必ず間違える。

- たとえば、hoyoyo.c が config.h を#include していると、hoyoyo.c を触らなくても config.h に手を入れると hoyoyo.c が変わったのと同じことになる。そんなことまで毎回判断するのは無理。

そこで、make は便利。

1.2 make の基本的な使い方

作業ディレクトリに makefile または Makefile というファイル名のファイルを作り，そこにルールを記述しておく。

make ターゲット

を実行すると、ルールにしたがってそのターゲットを作ろうとする。ターゲットを省略すると，Makefile に記述されている最初のルールのターゲットが補われる。

1.2.1 ルールの基本的な形式

```
ターゲット:  ソース ソース ...
        コマンド
        コマンド
          ⋮
```

ターゲットが存在しないかソースのどれかよりも古ければ、コマンドを実行する。依存関係も配慮して判断する。

■すべての規則をいちいち書く

```
# ayaya does not stand for Matsuura Aya

all: ayaya

main.o: main.c hoyoyo.h nahaha.h config.h
        cc -O2 -c main.c

hoyoyo.o: hoyoyo.c hoyoyo.h nahaha.h config.h
        cc -O2 -c hoyoyo.c

nahaha.o: nahaha.c nahaha.h config.h
        cc -O2 -c nahaha.c

ayaya: main.o hoyoyo.o nahaha.o
        cc -s -o ayaya main.o hoyoyo.o nahaha.o -lm

clean:
        rm -f *.o
```

- #から行末まではコメントであり、make からは無視される。人間のメモのために使う。
- 各コマンドの前にはタブを置く。スペース 8 個ではない。

- all と clean はダミーのターゲット。実際にその名前のファイルを作ることを意図していない。

■変数を使って変更しやすくする

```
# ayaya does not stand for Matsuura Aya

CC=cc
CFLAGS= -O2
LDFLAGS= -s
LDLIBS= -lm
OBJS= main.o hoyoyo.o nahaha.o

all: ayaya

main.o: main.c hoyoyo.h nahaha.h config.h
        $(CC) $(CFLAGS) -c main.c
hoyoyo.o: hoyoyo.c hoyoyo.h nahaha.h config.h
        $(CC) $(CFLAGS) -c hoyoyo.c
nahaha.o: nahaha.c nahaha.h config.h
        $(CC) $(CFLAGS) -c nahaha.c
ayaya: $(OBJS)
        $(CC) $(LDFLAGS) -o ayaya $(OBJS) $(LDLIBS)

clean:
        rm -f *.o
```

■特殊な make 変数（抜粋）

$@ ターゲット

$< ソースの一つめ

■サフィックスルールを使って楽をする

```
# ayaya does not stand for Matsuura Aya

all: ayaya

.SUFFIXES: .c .o

.c.o:
        cc -O2 -c $<

main.o: main.c hoyoyo.h nahaha.h config.h
hoyoyo.o: hoyoyo.c hoyoyo.h nahaha.h config.h
nahaha.o: nahaha.c nahaha.h config.h
ayaya: main.o hoyoyo.o nahaha.o
        cc -s -o ayaya main.o hoyoyo.o nahaha.o -lm

clean:
        rm -f *.o
```

■make 変数とサフィックスルールを組み合わせて使って楽をする

```
# ayaya does not stand for Matsuura Aya

CC=cc
CFLAGS= -O2
LDFLAGS= -s
LDLIBS= -lm
OBJS= main.o hoyoyo.o nahaha.o

all: ayaya

.SUFFIXES: .c .o

.c.o:
        $(CC) $(CFLAGS) -c $<

main.o: main.c hoyoyo.h nahaha.h config.h
hoyoyo.o: hoyoyo.c hoyoyo.h nahaha.h config.h
nahaha.o: nahaha.c nahaha.h config.h
ayaya: main.o hoyoyo.o nahaha.o
        $(CC) $(LDFLAGS) -o $@ $(OBJS) $(LDLIBS)

clean:
        rm -f *.o
```

■make 変数とデフォルトのサフィックスルールを使って、もっともっと楽をする

```
# ayaya does not stand for Matsuura Aya

CC=cc
CFLAGS= -O2
LDFLAGS= -s
LDLIBS= -lm
OBJS= main.o hoyoyo.o nahaha.o

all: ayaya

main.o: main.c hoyoyo.h nahaha.h config.h
hoyoyo.o: hoyoyo.c hoyoyo.h nahaha.h config.h
nahaha.o: nahaha.c nahaha.h config.h
ayaya: $(OBJS)
        $(CC) $(LDFLAGS) -o $@ $(OBJS) $(LDLIBS)

clean:
        rm -f *.o
```

2 発展

2.1 ちょっと進んだ使い方

2.1.1 make ルールの自動生成

多くの C コンパイラでは、-M オプションを使うと make ルールを自動生成できる。たとえば、

```
cc -M main.c hoyoyo.c nahaha.c
```

を実行すると、以下のものが、標準出力に出力される。

```
main.o: main.c /usr/include/stdio.h /usr/include/sys/cdefs.h \
  /usr/include/sys/_symbol_aliasing.h \
  /usr/include/sys/_posix_availability.h /usr/include/Availability.h \
  /usr/include/AvailabilityInternal.h /usr/include/_types.h \
  /usr/include/sys/_types.h /usr/include/machine/_types.h \
  /usr/include/i386/_types.h /usr/include/sys/_pthread/_pthread_types.h \
  /usr/include/sys/_types/_va_list.h /usr/include/sys/_types/_size_t.h \
  /usr/include/sys/_types/_null.h /usr/include/sys/stdio.h \
  /usr/include/sys/_types/_off_t.h /usr/include/sys/_types/_ssize_t.h \
  /usr/include/secure/_stdio.h /usr/include/secure/_common.h config.h \
  hoyoyo.h nahaha.h
hoyoyo.o: hoyoyo.c /usr/include/math.h /usr/include/sys/cdefs.h \
  /usr/include/sys/_symbol_aliasing.h \
  /usr/include/sys/_posix_availability.h /usr/include/Availability.h \
  /usr/include/AvailabilityInternal.h config.h hoyoyo.h nahaha.h
nahaha.o: nahaha.c /usr/include/stdio.h /usr/include/sys/cdefs.h \
  /usr/include/sys/_symbol_aliasing.h \
  /usr/include/sys/_posix_availability.h /usr/include/Availability.h \
  /usr/include/AvailabilityInternal.h /usr/include/_types.h \
  /usr/include/sys/_types.h /usr/include/machine/_types.h \
  /usr/include/i386/_types.h /usr/include/sys/_pthread/_pthread_types.h \
  /usr/include/sys/_types/_va_list.h /usr/include/sys/_types/_size_t.h \
  /usr/include/sys/_types/_null.h /usr/include/sys/stdio.h \
  /usr/include/sys/_types/_off_t.h /usr/include/sys/_types/_ssize_t.h \
  /usr/include/secure/_stdio.h /usr/include/secure/_common.h config.h \
  nahaha.h
```

GCC の場合、-M オプションのかわりに -MM オプションを使うと、システムヘッダファイルを除いたものが出力される。たとえば、

```
cc -MM main.c hoyoyo.c nahaha.c
```

を実行すると、以下のものが、標準出力に出力される。

```
main.o: main.c config.h hoyoyo.h nahaha.h
hoyoyo.o: hoyoyo.c config.h hoyoyo.h nahaha.h
nahaha.o: nahaha.c config.h nahaha.h
```

リダイレクションを利用して、これらをファイルに保存するようにすると、Makefile を作るのに便利。

2.1.2 インクルード機能

最近の make には、他のファイルをインクルードする機能がある。ただし、make の種類によってその構文が異なる。

GNU make の場合

```
include ファイル名 ...
```

BSD make の場合

```
.include <ファイル名>
```

または

```
.include "ファイル名"
```