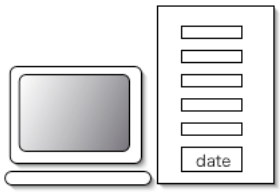
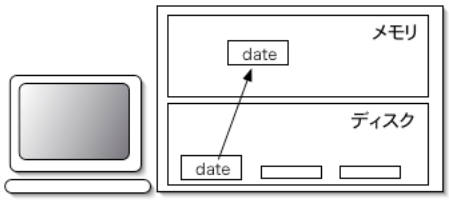
メモリとファイル

■ プログラムとファイルとメモリの関係



コンピュータの中には様々な機能が格納されており、それらには名前を付けられています。(CUI の説明でやりましたね)

具体的にはこれらの機能はディスクのなかにプログラムとして(ファイルという形で)格納されています。つまり date という名前をつけられたプログラムファイルがあるのです。

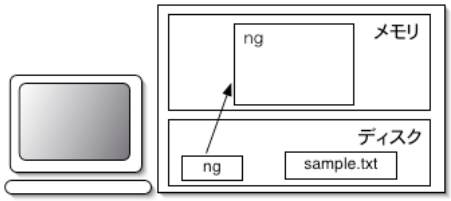


このプログラムを実行すると、メモリの中に読み込まれて、CPU がその機能を処理しはじめます。結果が画面に表示されたりするかもしれません。(されない場合もあります)

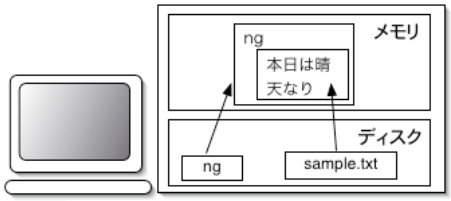
処理が終了すると、メモリ中に読み込まれたプログラムは廃棄されます。

■ エディタを起動してファイルを編集する、という動作の意味

ファイルを作成するために、またプログラムを編集(修正)するために、ng エディタを起動した場合、コンピュータの中では右図のような事が起こっています。
ng も date 同様にひとつのファイル、一つのプログラムですから、まず ng プログラムがメモリに読み込まれます。

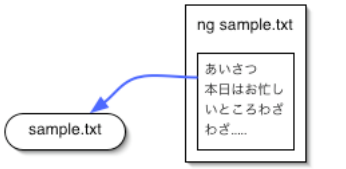


ng sample.txt としてファイルを指定してエディタ・プログラムを起動した場合、ng プログラムは起動後に sample.txt の中身をメモリの中に読み込み(引き写し)ます。



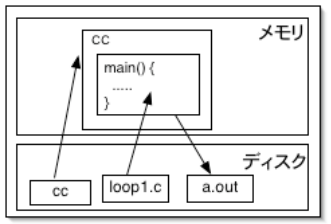
その後、メモリに読み込まれたファイルの中身を ng の機能を利用して編集(修正)するわけですが、それだけではファイルの中身は変化しません。(このときはメモリの中身が変化しているだけなので)

保存のための動作をして初めてファイルに結果が反映されます。
終了の時に保存するか否か尋ねられるのはこのためです。保存しないと返答すればメモリの中の編集結果は破棄されます。



■ cc コマンドでコンパイルする、という動作の意味

コンピュータが実行可能なプログラム、例えば ng コマンドや date コマンドは、人間(プログラマ)が直接記述するには非常に不便な、分かりにくい書き方で動作の指示が書き連ねられています。機械向きに設計された動作指示の言語(記述ルール)は人間には不向きなのです。(この種の言語を機械語と呼ぶ。)
それにくらべて C 言語によるプログラムは人間が読み書きしやすいように設計されており、通常はこうした人間向きに設計された記述ルールで動作指示を書きます。(この種の言語を高級言語と呼ぶ)



つまり人間向きの言語から機械向きの言語への変換を行う必要があるのです。cc コマンドは loop1.c という C 言語によるプログラムを、機械語のプログラムに変換し、a.out という名前で出力するコマンドです。(この種の機能のことをコンパイラと呼ぶ。 cc は C コンパイラにつけられた名前。)

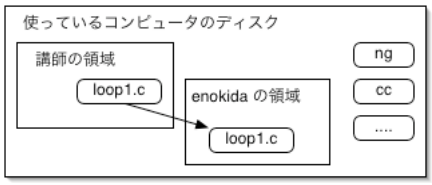
■ プログラムをコピーして、cc（コンパイル）して、./a.out（実行）するという動作の意味

先週行ったサンプルプログラムの実行手順について解説します。

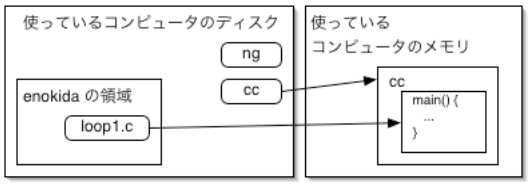
1. 講師のところからサンプルプログラム loop1.c を手元にコピーする。
2. loop1.c を cc コマンドでコンパイルする。
3. a.out を実行して動作を確認する。
4. loop1.c を ng エディタで開いて内容を修正する。
5. ふたたび 2. の動作、つまり cc コマンドでコンパイル。
6. 同じく 3. の動作、つまり a.out を実行して動作の変化を確認する。

以下にコンピュータの中で何が起きたのか説明します。

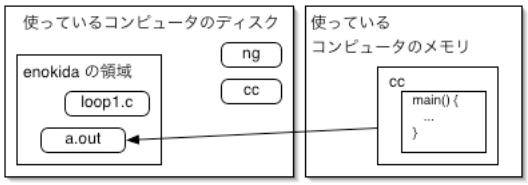
1. 講師のところからサンプルプログラム loop1.c を手元にコピーする。



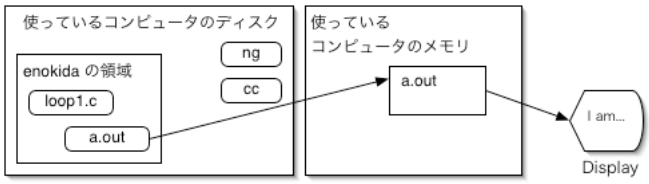
2. loop1.c を cc コマンドでコンパイルする。
まず cc プログラムが実行され、そこに loop1.c を読み込みます。



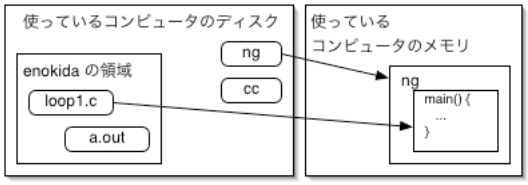
次に実行可能な形式に変換（コンパイルと呼びます）して、a.out という名前のファイルとして残します。



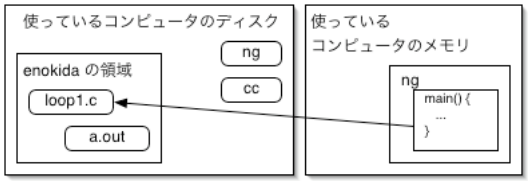
3. a.out を実行して動作を確認する。



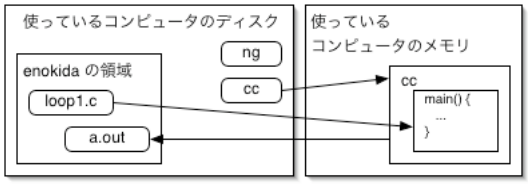
4. loop1.c を ng エディタで開いて内容を修正する。ng がメモリ上に loop1.c を読み込み、そこを修正します。



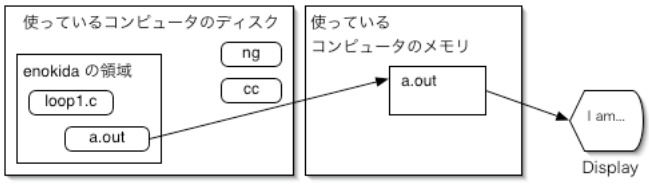
修正が済めば loop1.c に上書き保存、つまり loop1.c の内容が変更されたものに置き換わります。



5. ふたたび 2. の動作、つまり cc コマンドでコンパイルします。ccコマンドで loop1.c を読み込み、a.out に変換して保存します。



6. 同じく 3. の動作、つまり a.out を実行してプログラムを動作させます。このとき、loop1.c は修正が入っていますので、実行結果も変化しているはずです。それを確認しましょう。



■ 課題

先週の続きです。サンプルプログラムを講師の領域からコピーし、試しに実行し、修正を加えて再び実行し、修正結果が反映されることを確認して下さい。

まず原本をコピーします。以下のようにしてください。
```
cc2004(88%) cp /NF/home/kyoin0/yasuda/kisob/hello1.c  hello1.c
```
途中で TAB キーを使うとタイプミスを減らせます。または講師の教材 web ページからダウンロードすることもできます。http://www.kyoto-su.ac.jp/~yasuda/

hello シリーズは 1 から 4 まであります。順次試し、それぞれ「Hello World」などではない自分の好きなフレーズ（のようなもの）に置き換えて実行していってください。出力も一行だけではなく、複数行になるようにプログラムを書き換えて試すと良いでしょう。

hello1　から 4 まで修正できたら、前回同様に
1. まずdate コマンドで作業開始時間、
2. who コマンドで自分のユーザ名を表示させ、
3. hello1.c ~ hello4.c まで、それぞれ cat, cc, ./a.out を繰り返して画面に結果を出力、
4. 最後に date コマンドで作業終了時間を表示させる。
5. そして 1.~4. までの画面出力結果を先週と同じ方法（Copy & Paste）によってファイルに残し、
6. 印刷して提出。

押さえて欲しいポイント：

・行は「;」で区切られています。
・コメント（プログラムの実行に影響を与えないメモ領域）は /* ではじまり */ で終わる範囲です。
・{ } で囲まれる領域が一つのブロックを意味しています。
・for などのように（ ）で囲まれるパラメタ（引数）を意味する部分もあります。

いずれもプログラムの構造を分かりやすくするためのものです。

```
for(i=0; i < strlen(s); i+=1) {
    usleep(200000);
```

と書く方が、

```
for i=0  i < strlen(s)  i+=1
    usleep 200000
```

と書くより、どこからどこまでがどの要素（引数など）か、ということが分かりやすくなります。

つまりプログラムは右上例ではなくその下の例のように書く事も可能ですが、多くの場合は読みにくくなってしまいます。

```
/*
  473088 榎田裕一郎　経済学部 3 回生
  loop1.c
*/
main() {
  int i;
  char s1[256], s2[256];

  char *s="I am Y.Enokida";

  for(i=0; i < strlen(s); i+=1) {
    usleep(200000);
    printf("%c¥n",s[i]);
  };

}
```

```
/* 473088 榎田裕一郎　経済学部 3 回生
loop1.c  */ main(){int i;char s1[256],
s2[256]; char *s="I am Y.Enokida";
for(i=0; i<strlen(s); i+=1)
{usleep(200000);printf("%c¥n",s[i]);
};}
```

c 言語の文法は後で細かく教えますが、ともかくいまは全体的な構造の書き方があって、それに慣れる事が重要だと考えて下さい。