ソフトウェア演習２B

ライブラリの作成

ライブラリ

- 関数をまとめたもの
  - その言語で標準に提供されるライブラリや目的ごとに作成されて配布されているライブラリなどがある。
  - コンパイルされたバイナリ形式のライブラリとテキスト形式のライブラリがある

- バイナリ形式のライブラリには以下の2種類のライブラリが存在する
  - 静的リンクライブラリ
  - 動的リンクライブラリ

静的リンクライブラリ（static link library）

- プログラムを生成する際にライブラリをプログラムに組み込む（リンクする）タイプのライブラリ
- ファイルの拡張子は .a
- ライブラリの内容がプログラムに組み込まれているので、リンクしたあとでライブラリがなくなってもプログラムは動作する
- 利点：
  - 一度リンクしてしまえばプログラムのみで実行可能
  -
- 欠点：
  - ファイルサイズが大きくなる
  - ライブラリが更新された場合にプログラムをコンパイルし直さないといけない

動的リンクライブラリ（dynamic link library）

- プログラムを実行する際にライブラリをプログラムに組み込む（リンクする）タイプのライブラリ
- ファイルの拡張子は .so
- プログラム実行時にプログラムにリンクされるので、プログラムの実行時にリンク指定したライブラリが存在しないとプログラムが動作しない
- ライブラリの存在する場所が環境変数に設定されていないとプログラムの実行時にエラーになる
- 利点：
  - ファイルサイズが小さくなる
  - ライブラリが更新されてもプログラムをコンパイルし直すことなく実行することができる
- 欠点：
  - 関数の呼び出しに時間がかかる
  -

静的リンクライブラリの作成

1. ソードコードとヘッダーファイルの用意
2. オブジェクトファイルの生成
   - g++の-c オプションを使ってソースコード毎にオブジェクトファイルを生成する

     % g++ -c source1.cpp

     % g++ -c source2.cpp

3. ライブラリの作成
   - arコマンドを使ってライブラリを作成する

     % ar rcs libhogehoge.a source1.o source2.o

動的リンクライブラリの作成

1. ソードコードとヘッダーファイルの用意
2. オブジェクトファイルの生成
   - g++の-c オプションを使ってソースコード毎にオブジェクトファイルを生成する

     % g++ -c source1.cpp

     % g++ -c source2.cpp

3. ライブラリの作成
   - g++の −shared オプションを使ってライブラリを作成する

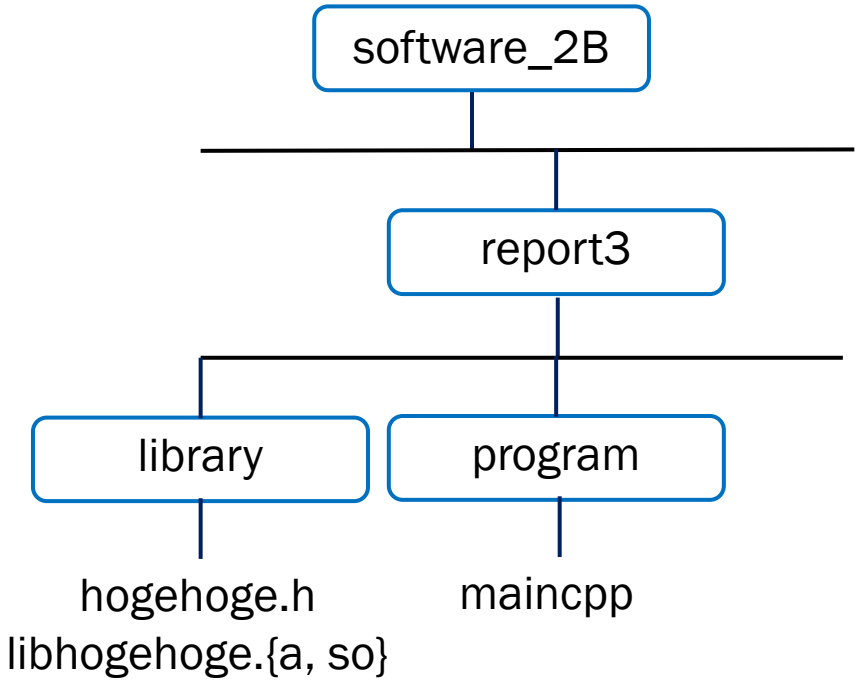
     % g++ -shared –o libhogehoge.so source1.o source2.o

ライブラリを使ったプログラムのコンパイル

- g++の-Iオプションと-Lオプションと-lオプションを使ってライブラリをリンクする
  - -I オプション：ヘッダファイルの場所を指定するオプション
  - -L オプション：ライブラリの場所を指定するオプション
  - -l オプション：ライブラリを指定するオプション

```
% cd program

% g++ main.cpp –I../library –L../library –lhogehoge –o program
```



プログラムの実行

- 静的リンクライブラリをリンクしたプログラムはそのままでプログラムを実行することができる
- 動的リンクライブラリをリンクしたプログラムはリンクしたライブラリの存在する場所を環境変数で設定しておかないと実行できない
  - ライブラリの場所は環境変数 LD_LIBRARY_PATH に設定する（シェルがbashの場合）

    % export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$HOME/software_2B/report3/library

  - $HOME/.bashrc に上記の内容を記述しておく（シェルがbashの場合）