# プログラミングC演習報告書
# プログラミングCレポートの書き方

【担当教員】　槇原 靖、小泉 佑揮　教員

【提出者】　阪大 太郎　(99990001)
ソフトウェア科学コース・2年
pro-c-as@ics.es.osaka-u.ac.jp
【提出日】　2014年4月1日

# 1　課題内容

レポート課題の問題文のまる写しは減点対象である。必ず、自分なりに書くこと。

# 2　プログラム全体の説明

本章では、プログラム全体の説明を記述する。実装したそれぞれの機能の説明は、次章以降で機能ごとに説明すること。

## 2.1　シェルの仕様

各自が作成したプログラムの仕様を書く。ここでは、各自が実装したシェルが持つ機能を説明する。各機能の具体的な説明は、後述の章で記述するため、ここに記述する必要はない。

## 2.2　処理の流れ・実装方法

ここでは、例えば、以下のような項目を記述する。

- ユーザーからの入力の処理
- 各機能（コマンド）への分岐の流れ

もちろん、これ以外にもプログラム全体で必要な説明を記述しても良い。また、必要に応じて「`\subsection{処理の流れ}`」、「`\subsection{実装方法}`」などを使って章を分割しても良い。

# 3　外部コマンド実行機能

## 3.1　仕様

各機能の仕様を記述する。課題文に記載されていない事項で、各自で定めた事項などもここに記述して良い。

## 3.2　処理の流れ

ここには、この機能を実現するための手順や方法、または、処理の流れを記述する。基本的にはプログラミング言語に非依存に記述されるべきである。例えば、ここには、以下のような内容を記述する。

> 子プロセスを作成し、その子プロセスをユーザーから指定された外部コマンドで置き換えることで、外部コマンドの実行を実現する。バックグラウンド実行かフォアグラウンド実行かは、このときの子プロセスの終了を待つかどうかで変更する。具体的には、フォアグラウンド実行の場合は子プロセスの終了を待ち、バックグラウンド実行の場合は、子プロセスの終了を待たずに即座にプロンプトを表示する。これらの処理は実装した exec_command 関数で実施する。

### 3.3 実装方法

ここには、この機能を C 言語でどのように実装したかを記述する。すなわち、自身の作成した関数、定義したデータ構造、ライブラリの呼び出しなどもここで説明する。例えば、ここには、以下のような内容を記述する。

> exec_command 関数では、まず、fork により子プロセスを生成する。作成された子プロセスを、execvp 関数によって、ユーザーから指定された外部コマンドに置き換える。execvp 関数を用いた理由は、XXX である。コマンドをフォアグラウンド動作させる場合には、wait により子プロセスの終了を待つ。フォアグラウンド動作かバックグラウンド動作かは、2 章で説明した parse_command で解析済みである。ここでは、その返り値に応じて、子プロセスの終了を待つかどうかを判断する。

「処理の流れ」、「実装方法」を必ずしも 2 つの章に分割する必要はないが、まず、極力プログラム言語に依存しない形式で処理の流れを説明し、その後、C 言語でその機能をどのように実装したかを記述するように心がけること。

### 3.4 テスト

#### 3.4.1 テスト方法

プログラムの動作テストは、作成したプログラムが正しく動作するかどうかをチェックするためのものである。従って、レポートではなぜそのテスト項目で動作チェックとして十分であるかという点を記述しなければならない。

#### 3.4.2 テスト結果

単に動作結果を示すだけではなく、簡単でもよいのでその結果から何が言えるのか、もしくは何が保証できるのかを書くこと。当然ではあるが、「動作することを確認した」だけでは、動作したことを保証できていないので、評価対象にはならない。

## 4 ディレクトリの管理機能

### 4.1 仕様

ディレクトリの管理機能では複数のコマンド（cd、pushd、dirs、popd）を指定している。このように、複数の機能がある場合、各コマンドを以下のように章ごとに記述しても良い。以下、「処理の流れ」、「実装方法」、「テスト」の章も同様である。なお、章に分割しない場合も、指定されたそれぞれの機能が正しく実装されていることを説明すること。

#### 4.1.1 cd コマンド

#### 4.1.2 pushd コマンド

#### 4.1.3 dirs コマンド

#### 4.1.4 popd コマンド

### 4.2 処理の流れ

### 4.3 実装方法

### 4.4 テスト

#### 4.4.1 テスト方法

#### 4.4.2 テスト結果

## 5 ヒストリー機能

ヒストリー機能群に関する説明を記述する。記述する内容は、これまでの章と同様に、「仕様」、「処理の流れ」、「実装方法」、「テスト」の項目とする。また、ヒストリー機能の拡張を実装した場合は、この章にその内容を記述する。

なお、本章に限らず、既に説明した機能をここでも利用する場合や、既に説明した内容と説明する内容が重複する場合などは、「AAA の機能は 3 章の XXX を応用する」「YYY の処理は、3 章の ZZZ の処理と同様である」など、当該箇所を適宜参照すること。また、他の文献を参照した場合は、「XXX の実装方法については、文献 [1] を参考にした」のように、参考にした文献を明示的に記述すること。参考文献は、レポートの最後の章にまとめること。

## 6 ワイルドカード機能

ワイルドカード機能群に関する説明を記述する。記述する内容は、これまでの章と同様に、「仕様」、「処理の流れ」、「実装方法」、「テスト」の項目とする。

また、ワイルドカード機能の拡張を実装した場合は、この章にその内容を記述する。

## 7 プロンプト機能

プロンプト機能群に関する説明を記述する。記述する内容は、これまでの章と同様に、「仕様」、「処理の流れ」、「実装方法」、「テスト」の項目とする。

## 8 スクリプト機能

スクリプト機能群に関する説明を記述する。記述する内容は、これまでの章と同様に、「仕様」、「処理の流れ」、「実装方法」、「テスト」の項目とする。

# 9　エイリアス機能

エイリアス機能群に関する説明を記述する。記述する内容は、これまでの章と同様に、「仕様」、「処理の流れ」、「実装方法」、「テスト」の項目とする。

# 10　自分で考えた機能

指定した機能を少しだけ変えたものを「自分で考えた機能」とした場合の評価は低くなる。機能自体を自身で考えること。たとえば、ヒストリーを逆順で表示する「rhistory コマンド」を考えたとする。これは、自分で考えた「機能」ではなく、単なる history コマンドの異なる表示方法を考えただけである。機能発案の創造性を問う課題ではないので、有用な機能である必要はなく、また、既存のシェルに実装されている機能を禁止するものでもない。課題で指定されたコマンドの亜種ではなく、自身で機能を考えることが重要である。

なお、自分で考えた機能については、レポート中にコマンド名も明確に記述すること。

# 11　その他実装した機能

指定された機能以外にも何か実装した場合は、このような章を用意しても良い。

## 11.1　仕様

コマンド名も記述すること。

## 11.2　処理の流れ

## 11.3　実装方法

## 11.4　テスト

### 11.4.1　テスト方法

### 11.4.2　テスト結果

# 12　工夫点

プログラム作成で、工夫した点を自由にアピールしてよい。ただし、本当に工夫したことを正しく伝えること。たとえば、「使用するメモリを削減した」のような記述を良く見るが、これでは工夫点として評価できない。具体的に、どこをどのように工夫したかなど、自分の工夫を教員に伝える努力をすること。なお、さすがに「変数名をわかりやすくした」程度では評価対象にはならないので、評価に値する内容を記述すること。

## 13　考察

作成したシェルに関する考察を、客観的に記述する。「〜だと思う」のみでは、主観的な記述であるので、それを裏付ける根拠や普遍的な妥当性も同時に記述すると良い。

## 14　感想

課題の感想、苦労した点、講義や演習に対する要望など自由に書いてよい。

## 15　謝辞

必要に応じて記述する。

## 参考文献

[1] B. W. Kernighan and D. M. Ritchie, “プログラミング言語 C,” 第 2 版, 共立出版, 1989.

# A プログラムリスト

```
#include <iostream>

int main(int argc, char *argv[])
{
    int sum = 0;
    for(int i = 0; i < 10; ++i) {
      sum += i;
    }
    std::cout << sum << std::endl;
    return 0;
}
```