はじめに

授業概要

情報システムを駆動する元になっているシステムサービス（ＯＳを含む）を動かすためには，システム全体を見据えたプログラミング能力が必要である。この実習では，プログラミング言語をある程度習得した学生を対象として，スクリプト言語を使用したシステムの自動処理や，正規表現等の知識表現手法に基づいた自動処理の設計，多言語混在環境での開発手法，ネットワークサービス等の開発に必要な技術，並列プログラミング，ハイパフォーマンスコンピューティング，グリッド・コンピューティング技術など，実用的なシステムサービス，問題解決システムの構築に役立つ技術の習得を目標としている。また，本実習は単独のプログラミング実習という位置づけではなく，専門演習，卒業研究での円滑な研究活動のサポート実習となることを目指す。

授業計画

1. 実習計画，課題の提出方法，成績評価の方法の確認・システムプログラミングの基礎
2. スクリプト・プログラミング
3. システムサービスの起動プロセス
4. コンピューティングシステムの実行
5. 分散開発手法
6. 多言語混在開発手法
7. システム構築の自動化
8. TCP/IP サービスの基礎
9. ネットワークサービスの構築・実行
10. プログラムの並列実行
11. PVM によるプログラムの並列実行
12. MPI によるプログラムの並列実行
13. 応用問題への取り組み (1)
14. 応用問題への取り組み (2)・まとめ

成績評価の方法

定期試験を行わず，平常試験（小テスト・レポート等）で総合評価する（レポートの具体例は，後述する「**レポートの提出について**」を参照）．出席点，課題の成績から総合的に判定する．おおむね 50:50 で配分する．尚，4 回以上の欠席，課題未了の者は単位認定を行わないものとする．

毎回，難易度を変えた課題を設定する．
どのレベルの課題まで回答しても構わないが，評価の最大値は課題のレベルによる．回答した課題はまとめて提出すること．あとから追加して提出されたものは，添削はするが成績評価の対象外とする．やはり，回答全体を遅れて提出した場合は，大幅減点にせざるをえない．
ほとんどの受講生が「課題（竹）」に取り組むことを期待している．それぞれの難易度で評価点数の上限を設定しているが，その課題を提出すれば上限となる点数が保証される，ということではない．課題（竹）を提出しても，評価が 60 点を下回ることもありうる．
それはなぜか？それは提出されたレポートの質による．いい加減に執筆されたレポートは点数は低くなるし，しっかり考えて作られたレポートは上限に限りなく近い評価をする．「いい加減」というのは，実行結果から必要なものを抜き出さずに，ターミナルに出てきたものを繰り返し何度も貼り付ける，とか，コピペのレポートを作成する，とかのことである．
この実習では，その辺りの評価も行う．専門演習・卒業研究の実験レポートや卒業論文等で役に立つことと思う．

課題（梅）まず，課題（梅）に回答してもらう．評価の上限は 60 点とする．実習資料で説明している内容の確認程度の課題なので，課題（竹）への足がかりとして取り組んでもらいたい．

課題（竹）課題（梅）に取り組むまでもない受講生は課題（竹）から始めてもらって構わない．（梅）を回答しているかどうかにかかわらず，評価の上限は 89 点とする．

課題（松）（竹）の回答を必須として，最高 100 点とする．評価が 90 点を越えるには，（竹）の回答とこの課題に取り組む必要がある．

課題提出のルール

課題の提出は，次回の実習開始直前までに，関大 LMS へのオンライン提出とする．

- 提出期限までの間に再提出が必要になった場合（間違いを見つけた，等）は，担任者に申し出ること．

- 提出期限日・時刻を過ぎたものの提出は，印刷物の提出のみとする．ただし，実習時から 2 週間を超えての提出は認めない（つまり，すべての課題を提出できない＝成績評価の対象外．これはシラバスの成績評価方法にあるとおり.）．遅れて提出したレポートは，難易度を問わず，評価の上限を 60 点とする．
- 特別な事情がある，またはあった場合については，予め担任者に連絡し，相談すること．
- 最終回および最終回の前の回の取り扱い

  最終回および最終回の前の回については，最終回の実習時間内をもって最終受付とする．特別な事情がある場合は，予め担任者に連絡すること．

教科書

配布プリントにより実習を行う．配布方法については第 1 回で説明する．

参考書

シラバスに掲載しているとおり．適宜，追加する．

備考

実習はすべて iMac 端末で実施する．Unix の基本（ファイル，ディレクトリ管理，エディタ，コンパイラ，ブラウザ利用等）を理解している前提で実習を行う．

レポートの提出について

この実習で要求するレポートは，完成度の高いレポートである．単位認定の前提条件として，すべての課題を提出してもらわなければならないが，提出するだけで満点が保証されるわけではない．中途半端なものはバシバシ減点していく．

毎回，課題の種類は異なっており，プログラムの作成，プログラムの実行結果など，さまざまである．それらをきちんと**まとめて**レポートとして執筆すること．まとまっていない，実行結果をいい加減に載せている，動かないプログラムを載せている，などのレポートは減点していく．

レポートの執筆方法は，pLaTeX 2_ε で清書，Word または Pages で清書の 2 通りある．

pLaTeX 2ε で清書

pLaTeX 2ε 版の雛形ファイルは，/public/Styles ディレクトリ内の prac-blank.tex である．このファイルは，必要最低限のものしか入れていない，ほとんど空（から）のファイルになっている．文字コードが UTF-8 になっているので，PC 版や Mac 版の pLaTeX 2ε で清書する際は，UTF-8 対応になっているかどうかを確かめてほしい．もしも，Shift-JIS 版のものを利用したい場合は，同じディレクトリの prac-sjis.zip に含まれているので，それを利用してほしい．

以下，PDF ファイルを作成するまでの手順である．

1. fireport.sty を作業場所にコピーする．

```
user@tc2301<1> mkdir  -p   適切なディレクトリ ↵
user@tc2301<2> cd  適切なディレクトリ ↵
user@tc2301<3> cp /public/Styles/fireport.sty  ./  ↵
```

2. prac-blank.tex を適切な名前でコピーする．ここでは prac.tex とする．

```
user@tc2301<3> cp /public/Styles/prac-blank.tex  ./prac.tex  ↵
user@tc2301<4> chmod u+w ./prac.tex ↵
```

   同じファイルを別の課題に再利用するようなことはせず，毎回別のファイルに保存すること．実習 1 回分の課題は同一ファイルにすること．

3. prac.tex を編集して，レポートを執筆する．

4. prac.tex をコンパイルして，DVI ファイル（prac.dvi）を作成する．

   この段階でエラーが出る場合は，修正すること．エラーがでていると DVI ファイルは絶対に作成できない．

```
user@tc2301<1> platex  prac.tex ↵
```

5. DVI ファイルを PDF ファイルに変換する．

```
user@tc2301<1> dvipdfmx  prac.dvi ↵
```

6. 出来あがった prac.pdf を表示・確認・印刷する．

```
user@tc2301<1> open prac.pdf ↵
```

   仕上がりを十分確認の上，PDF ファイルをレポート提出システムで提出する．

Word または Pages で清書

/public/Styles ディレクトリには，Word または Pages 用のレポート向けテンプレートを用意している．Word 用は fireport.docx，Pages 用は fireport.pages である．

```
user@tc2301<1> cp /public/Styles/fireport.pages  ./ex01.pages ↵
user@tc2301<2> chmod u+w ./ex01.pages ↵
user@tc2301<3> open ex01.pages ↵
```

LaTeX が使いにくければそれらを使って執筆してもらいたい．

VIII

レポート作成の注意点

レポートを作成するにあたっては，内容を十分に精査すること．ギリギリになって適当にプログラムリスト，実行結果などを貼付けているようでは間に合わない．

内容に関する注意点

- レポートして記述すべき内容を理解しているか，
- レポートとして十分な内容になっているか，
- 必要事項は記述されているか，
- 不要な文言，行，プログラム等が含まれていないか，
- 期限までにできない，わからない課題について，「できない」「わからない」という報告のみになっていないか，
- 他人のレポートを写していないか，デジタルコピー，コピー&ペーストだけでなく，タイピングによる引き写しも含めて．

文書構造に関する注意点

- それぞれの課題の区別ができているか，節見出し等が適切に付けられているか，
- 文章，図，プログラムリスト，実行結果等，それぞれが区別して記述されているか，
- 図表を入れる場合，図表番号，見出しが適切に付けられているか，
- 図表を入れる場合，本文での言及を伴っているか，

「レポートの締切は次回の実習開始まで．やむを得ない事情で遅れて提出することは認めるが，当該実習の 2 回あとの実習開始時まで．『やむを得ない事情』には，『他の講義・実習のレポートで忙しい』『アルバイト・他の用務で時間が取れない』『理解が及ばない』『レポートが完成しない』等は含まないものとする．」

ちなみに，本実習のレポート提出ルールは，年々条件が付け加えられており，どんどん厳しくなっていることを付け加えておく．これ以上条件が付け加えられないよう，今年度の受講生のレポート提出状況に期待する（これ以上増えては，来年度の受講生が可哀想である）．

システムプログラミング実習　第1回

システムプログラミングの基礎

ある情報システムを構築するためには，コンピュータの上で動かすためのプログラミングが少なからず必要である．そのシステムは，ある一定の数式に基づいた計算プログラムかもしれないし，不確定な数の要素を使って繰り返し実行が必要なプログラムかもしれない．また，単純に 1 台のコンピュータで実行すると膨大な時間がかかってしまうプログラムかもしれない．

それぞれの問題を解決するために，プログラムを作成し，実行して解を得る，というプロセスは同じであるが，問題に適したプログラム，プログラムの構築方法というものも存在する．ある一定のパターンを別のパターンに変換するだけなら，何も独自のプログラムを作成する必要もなく，パターン処理に適したプログラムで実行すれば良い．リスト処理が必要ならば，リスト処理を得意とするプログラミング言語を使えばよい．システムを起動する際に，あるコマンドの実行結果の一部分を使って，別のコマンドを実行していく，ということをしなくても，決まった形で実行できるのなら，スクリプトを作成してそのスクリプトで自動的に実行した方がはるかに効率よくシステムの起動ができる．とても大きな問題を分割して個別にプログラムを組まないでも，並列コンパイラや分散処理環境を使えば，ほとんど自動で並列分散処理をやってくれる．

コンピュータは「人の作業を楽にするモノ」であって,「人が苦労するモノ」ではないのである．この実習では，苦労しないですむ方法についても講義，実習を行っていく．

1.1　プログラムの作成

人の代わりにコンピュータに仕事をしてもらうのであるが，人がその仕事の流れを知らなければ，コンピュータに仕事を依頼することさえできない．その仕事の流れがフローチャートであり，アルゴリズムである．

単純に問題解決のためのシステムであれば，問題解決のアルゴリズムを実装し，そのプログラムを実行すればそれで済む．しかし，問題解決のためだけでなく，ネットワーク上のクライアントにサービスを提供するものであると，一度に一つのクライアントにサービスを提供するだけでは不十分であるので，事情は異なってくる．これは並列分散型のプログラムでも同様で，並行して，または分散して動作するプログラム間の相互作用を考慮する必要が出てくる．

また，実装可能性を検証するだけならば，すべての機能を一つのプログラムで実装する必要はない．もちろん，そのまま正式版のプログラムとなることもある．この場合には，さまざまな言語で作成されたプログラムが混在することもあり得る．それは，それぞれの言語に得意な処理，不得意な処理があるため，複数の言語を使ってそれを補って，システムを構築していくことになる．

1.2　プログラムの起動

計算用のプログラムでは，必要な計算過程がプログラミングされているので，ただ単にコマンドによって，プログラムを起動すればよい．しかし，汎用性を持たせたプログラムにすることを考えると，対話型のプログラム，起動オプションによって挙動が変化するプログラムになってくる．

今，カレントディレクトリにある command というコマンドを実行するためには，以下のように実行する．

```
unix% ./command ↵
```

コマンドを./で始めるのは，カレントディレクトリにある command を起動するということを明示的に示すためである．./をつけなかった場合は，コマンド検索パス[1]の中から探して実行されるため，カレントディレクトリのものが選ばれるとは限らない.[2]

コマンドには起動オプションというものが備わっていることがほとんどである．コマンドの実行時に同時に指定するもので，例えば ls コマンドであれば，

```
unix% ls ↵
```

によってファイルの一覧が表示される．もっと詳しい情報がほしければ，

```
unix% ls  -l ↵
```

のように，-l オプションをつけて実行すると，表示される内容が変化する．起動オプションとは，このように通常隠されている情報を表示したり，逆に必要な情報に絞り込んだりすることができるものである．この起動オプションの機能を自作プログラムに組み込むと，一々別のプログラムを作らなくても，いろいろな機能を持ったプログラムを開発することができる．

対話型のプログラムでは，プログラムのメッセージに対して，ユーザーが適当な入力を行ってプログラムの動作を継続する．プログラムの作りにもよるが，整数型を要求している箇所に実数型を入力するとプログラムの動作がおかしくなる場合もあるので，入力されたものが正しいかどうかについても検証する必要がある．このチェックはユーザープログラムで行う必要がある．

起動オプションによってプログラムの挙動を変える場合，対応しているオプションかどうか，オプションに指定されたパラメータが正しいかどうかについて検証する必要がある．前者についてはライブラリ関数で実現できているので，後者をユーザープログラム側でチェックすればよい．

ライブラリ関数の利用方法については，オンラインマニュアル以外に，ソースコードの配布されているフリーソフトウェアがよい教材となる．プログラミング上のテクニック等

[1] C-shell 系であれば which，sh 系であれば type コマンドで，実行したいコマンドがどのパスに存在するかを確認できる．

[2] 意図しないコマンドが動くことになり，トラブルの原因にもなる．スーパーユーザーの場合には深刻な結果となることがあるので，実行しようとしているコマンドのパスには注意すること．

についてもふんだんに盛り込まれているので，いずれかのフリーソフトウェアの読解にチャレンジすることも，プログラミング技術の向上のためには必要である[3]．

対話型のプログラムでも，入力リダイレクションの機能を利用することで，その処理を自動化できる．もちろん，そのプログラムの内容を理解していることが前提である．次の例は，グラフを描画するコマンド gnuplot によって，自動的にグラフをファイルに保存するシェルスクリプトの例である．

```
#!/bin/sh

gnuplot << EOF
pi=3.141592653589
set grid
set xrange [-pi:pi]
set terminal postscript eps
set output 'sin.eps'
plot sin(x)
quit
EOF
```

同じようなグラフを描く場合，gnuplot のサブコマンドを何度も入力することでも実行できるが，それは労力の無駄遣いである．サブコマンド群をファイルに保存しておいて，それを入力リダイレクションで入力してもよいし，前述のシェルスクリプトに記述しておいて，それを実行してもよい．少ししか違わないグラフ，例えば $\cos(x)$ のグラフを描く場合には，保存しておいたシェルスクリプトや入力ファイル内の sin の部分を cos に変更し，グラフの保存ファイル名を cos.eps に変更するだけで，あとはこのスクリプトを実行すればよい．同じようなコマンドを何度も入力する必要はなくなるのである．

通常は，gnuplot のコマンドの指定だけで実行するが，そのコマンドに続けて<< 文字列を指定すると，次の入力から『文字列』が出現するまで，そのコマンドへの入力とすることができる．上の例では，gnuplot の次から EOF の前までが，gnuplot の入力として扱われる．

この例では gnuplot について示したが，入力リダイレクションを使う場合にはそのコマンドの動作を理解しておくことが重要である．

[3] Solaris8 のソースコードは
~kobayasi/EXAMPLES/sol8src/osnet_volume/usr/src/cmd 以下に展開しているので，参考まで．

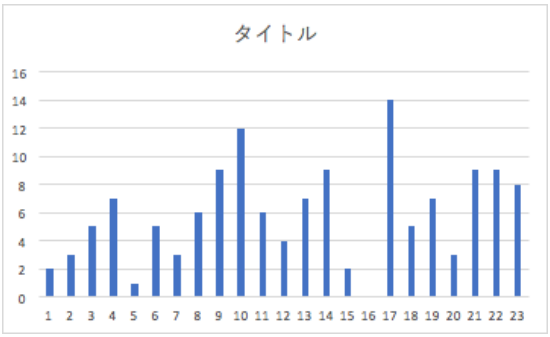

図 1.1: Excel でのグラフ（標準）

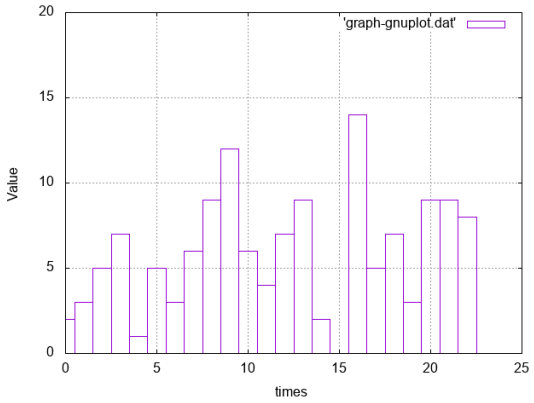

図 1.2: より適切なグラフ

同じグラフを描くにしても，Excel での標準的なグラフは使い物にならないことを理解すべきである．

このように Excel のグラフは作成しただけでは (1) 軸のタイトルがない，(2) 全体の「タイトル」という余計なものがついている，(3) 凡例がない，(4) 軸のメモリが適切でない等，かなり手を加える必要がある．「Excel が使える」とは，グラフについてもこういった設定をキチンとできる，ということであるので，Excel を使うなら精進してもらいたい．

1.3　サービスの起動

Unix 系 OS におけるサービスの起動には，古い作業ファイルの消去，二重起動のチェックなどの前処理に，さまざまなプロセスが関与する．また，終了時にも同様の手順が存在し，それらのコマンドを一つずつ手動で起動していたのでは，迅速なサービス提供，再開に間に合わない．

近年の OS におけるサービスの起動には，一連のプログラム，コマンドの実行順序を記述したシェルスクリプトが使われることがほとんどである．15 年ほど前までの Unix 系の OS では，すべてのサービスを起動するために一つのシェルスクリプトを使っていたが，最

近では，それぞれに個別のシェルスクリプトによって起動することが主流になってきた[4]．

```
unix# sh  /etc/rc.d/hoge  start ↵
```

のように起動する．停止の場合は，

```
unix# sh  /etc/rc.d/hoge  stop ↵
```

である．起動・終了方法はどの OS，サービスでもほぼ同じである．これらのシェルスクリプトの作成方法については後日行う予定である．

シェルスクリプトを作成するには，そのスクリプトで実行するコマンドの使い方や起動オプションをある程度習得する必要はある．習得しなくてもマニュアルを調べることによってどのような機能を持っているのか知ることができる．すべてを Web で調べるユーザーがかなりいると思うが，コマンドのマニュアル（「man コマンド名」と実行）することでも簡単に調べることができる．

コマンドやスクリプトの実行結果は，コマンドなどが正しく動作していれば何も出てこないことが多いので，実行した結果作成されるファイルなどを確認して確かめること．

1.4 セキュリティ対策

2005 年 4 月から個人情報保護関連法が全面施行になった．これらの法律では個人情報の保護についての配慮をせよ，ということが謳われている．情報システムに関しては，個人情報の保護だけでなく，情報システムを守ることも考えなければならない．そのためには，できるだけ安全に利用できるプログラムを書くことが必要になる．

少なくとも，使うこと自体が危険であると分かっているライブラリ関数は使用しない，入力される値を必ずチェックする，などのルールを守るだけでもかなり安全なプログラムとなる．Web の CGI や Web データベースで用いられているプログラミング言語（PHP や Perl など）ではこれらのルールに加えて，サーバ上の重要な情報が漏洩しないようにする措置を施さなければならない．

課題（梅）

ホームディレクトリ内から拡張子.docx を持ったファイルの一覧を作成し，その個数を数えて報告する．ファイルの一覧を作成するコマンド，行数を数えるコマンドを使用するとよい（ネットワーク実習の範囲）．

[4] 10 年前は/etc/rc.local，現在は/etc/rc.d や/usr/local/etc/rc.d にある起動スクリプトである．

課題（竹）

以下に挙げるコマンドの機能について調査し，どのようなオプション機能が備わっているのか，まとめなさい．5個以上のオプションを持つコマンドについては代表的なコマンドラインオプション5個について，5個未満のコマンドについては全部のオプションについてまとめること．オプションがないものについては，サブコマンドの説明を同様にまとめること．

`awk`，`grep`，`sed`，`cut`

オンラインマニュアル，Webページの引き写しは不可．

課題（松）

1. 3ページの`gnuplot`コマンドの内容をシェルスクリプトに入力し実行する．実行方法は「sh スクリプト名」である．このスクリプトの実行結果は`sin.eps`というPostScript形式のファイルであるので，そのファイルの内容を確認し，そのグラフを貼り付けること．このグラフファイルの品質を落とさずに貼り付けること（画面キャプチャは品質が落ちるので，使ってはいけない）．

2. 3ページの`gnuplot`のコマンド（4〜9行目）を手作業で入力し，グラフを描画するまでの時間を計って報告する．10回計測し，その平均を求めること．ストップウォッチ機能を持っていればそれで時間を計測する．持っていなければ，秒針で計測する．