シェルプログラミング入門　応用例

授業レポート自動受けとりシステムについて

山之上 卓[1]

1　はじめに

授業レポートの自動受けとりシステムを，シェルスクリプトと awk を組み合わせることによって作成した．このシステムは，電子メールで教官に送られてきたレポートを，教官のホームデイレクトリの下に整理して保存し，レポート受け取り通知を電子メールで学生に通知するものである（図 1）．

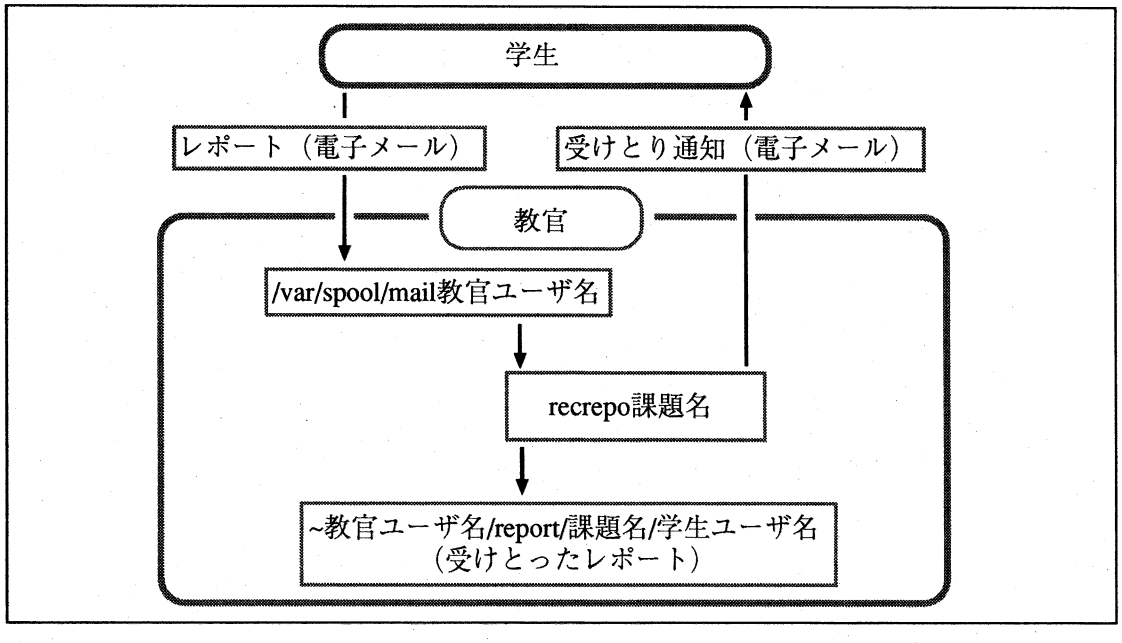

図 1: レポート受け取りシステムの概要

[1]情報科学センター，yamanoue@isct.kyutech.ac.jp

本システムは7つのシェルスクリプトから構成されているが，それぞれは2行から8行までの短いものである．本システムは作成に取り掛かってから2日程度で完成した．このシステムは実際の授業で使用しており，レポート整理の手間の軽減，紙資源の節約等に役立っている．また，受け取ったレポートは学生全員にも読めるようにしている（読めないようにすることも可能である）．良いレポートも悪いレポートも学生同士で見ることができるので，良い刺激になっているようである．学生がレポートをコピーして提出した場合，そのことは全学生にも直ちにあきらかになるため，かえってコピーは減少しているようである．

2 本システムの使用法

このシステムでレポートを受けとる場合は，以下のような手順を行なう．

1. 教官が，課題名（電子メールの Subject）とレポート提出期限を指定して出題する（図2）．このときの課題名は，”bin” 以外の，空白をはさまない英数字の列とする．

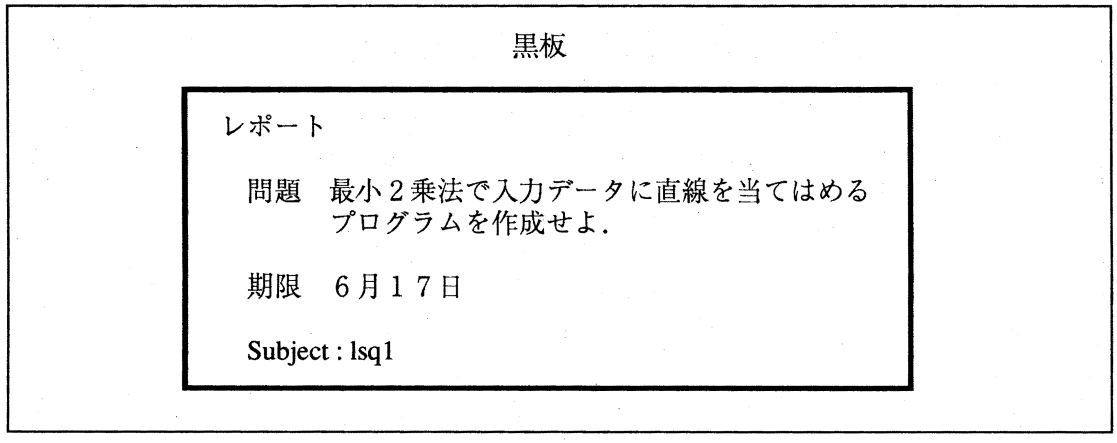

図 2: レポート課題出題例

2. 学生側でレポートを emacs 等で作成し，教官宛に電子メールで送る．このとき，電子メールの題目 (Subject) は，1. で教官が指定した 課題名を使う（図3）．なお，レポートの本文中に 「Subject: 課題名」で始まる行が存在すると，動作が保証されないので学生に注意しておく．

Emacs (M-x mail)

To: tyamano
Subject: lsq1
-text follow this line-

学籍番号 92XXXXX
名前　九州太郎
作成年月日　6月16日

問題
　　最小２乗法で入力データに直線を当てはめる
　　プログラムを作成せよ

プログラムとその解説
　　program lsq1
　real x{50},y{50}....入力データ格納配列
　　...
　　end

考察
　　連立一次方程式を解く部分にガウス掃きだし法を
　　利用できる.
　　この場合のプログラムは以下のようになる.
　　.....

感想
　　入力データと，あてはめた直線をグラフィクスで
　　表示させたい

図 3: レポート例

3. レポート提出期限が来たら，教官は

```
cd ~/report/bin
recrepo　課題名
```

を実行する．これによって，ファイル

~ 教官ユーザ名 /report/ 課題名 / 学生ユーザ名

にその学生のレポートが格納される（図 4）.

また，電子メールで図 5 のようなレポート受けとり通知が，学生に送られる.

4. 教官および学生が

`emacs ~教官ユーザ名/report/課題名`

等のコマンドを実行して提出されたレポートを確認する.

```
tyamano%    ~/report/bin/recrepo lsq1

tyamano%    ls-l   ~/recrepo lsq1
-rw-r--r--   1 tyamano      1593 Jun 16  09:18 x925xxyx
-rw-r--r--   1 tyamano      1582 Jun 16  09:18 x925xpyn
-rw-r--r--   1 tyamano      1756 Jun 16  09:18 x925xuay
....
```

図 4: レポート受け取りプログラムの実行

```
From: yamanoue takashi ⟨tyamano⟩
Message-Id: ⟨9206160743.AA08028@post.isct.kyutech.ac.jp⟩
To: x925xxyx
Date: Tue, 16 Jun 92 16:43:18 +0900

report lsq1 を受けとりました.
/home/tyamano/report/lsq1/x925xxyx にありますので確認してください.
```

図 5: レポート受け取り通知メール

3 レポート自動受け取りシステムの構成

本システムは `~/report/bin` の下に，受け取りシステムのシェルプログラムを格納している．受け取ったレポートは `~/report/課題名` の下に格納される.

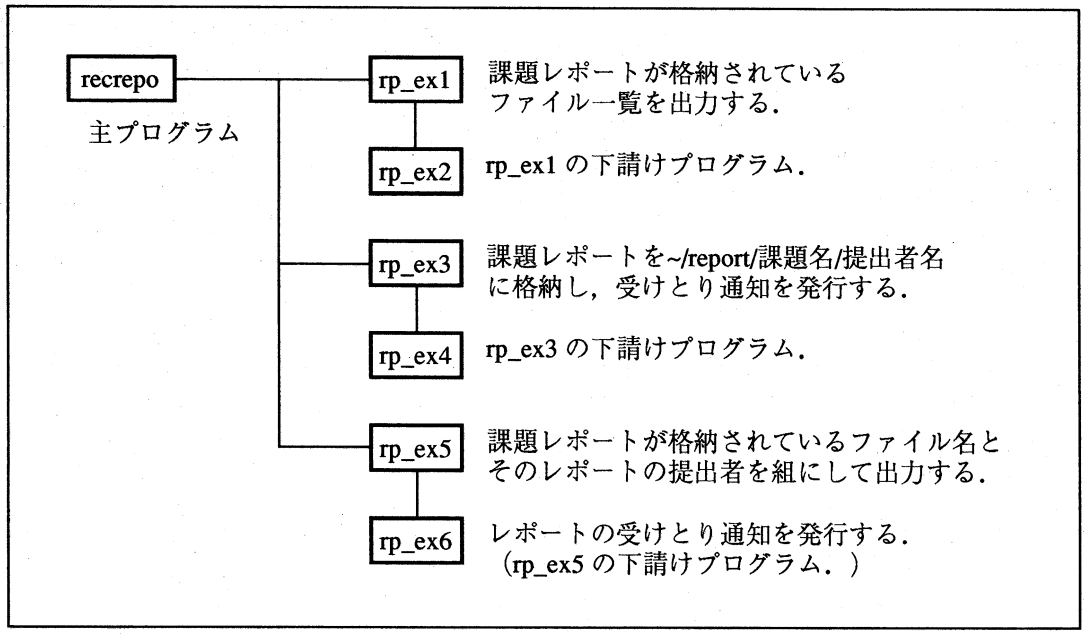

図 6: 本システムを構成する 7 本のプログラムの概要

本システムは図 6のような 7本のシェルプログラムによって構成されている.

以下に各プログラムの詳細を述べる.

3.1 recrepo

本システムの主プログラム. 第 1 引き数に課題名を指定して利用する. 図 7に内容を示す.

```
#!/bin/sh
inc
mkdir $HOME/report/$1
ls $HOME/Mail/inbox |sort -n|rp_ex1 $1|rp_ex3|rp_ex5 $1
```

図 7: recrepo

inc はメールスプールに格納されている未読のメールを `~/Mail/inbox` の下に格納する. このとき, 個々のメールは別々のファイルとなり, そのファイル名は正の整数である. `mkdir $HOME/report/$1` はディレクトリ `~/report/` 課題名 を作成する. もし既にこ

のディレクトリが存在していた場合は，その旨がメッセージとして示され，次のコマンドが実行される． `$HOME` はホームディレクトリ(`~/`)を表す環境変数である．"`$1`" はこのシェルプログラムの第一引き数（この場合は課題名）を表す． `ls $HOME/Mail/inbox` は `~/Mail/inbox` に格納された電子メールのファイル一覧を出力する．このコマンドの右には，パイプを表す"`|`" があるため，このコマンドの出力は，画面には表示されず，"`|`" の右にあるコマンド"`sort -n`" の入力となる．従って，"`ls $HOME/Mail/inbox |sort -n`" は電子メールのファイル一覧を，昇順に並び変えて出力することを表す．

この出力はパイプによって次の"`rp_ex1 $1|rp_ex3|rp_ex5 $1`" に引き渡される．なお，パイプで結合されたコマンドのことを「フィルター」と呼ぶ．フィルターの列"`rp_ex1 $1|rp_ex3|rp_ex5 $1`" には図8のようなデータが流れる．

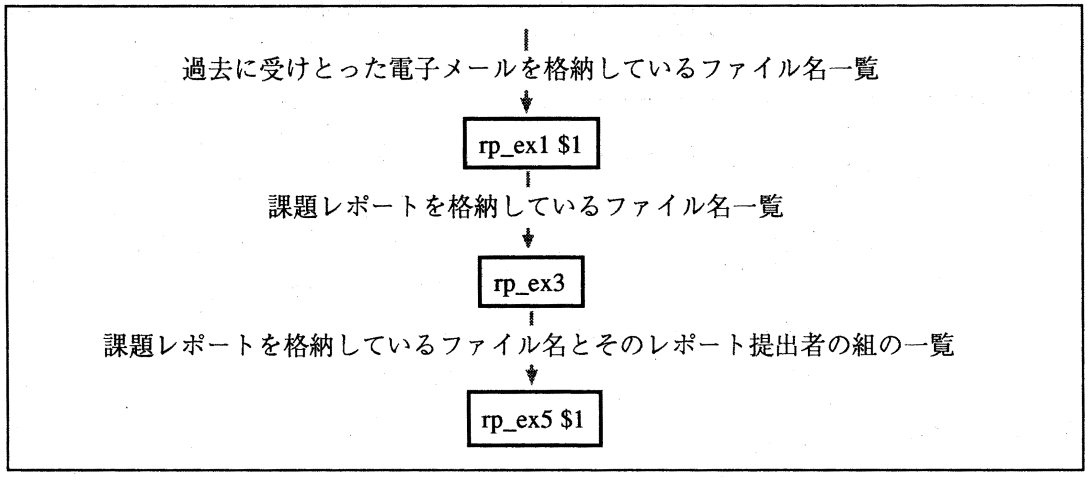

図 8: recrepo 内のフィルターの列を流れるデータ

3.2 rp_ex1

パイプ（または標準入力）から電子メールを格納しているファイル名一覧を入力して，課題レポートを格納しているファイル名一覧を出力する． 第一引き数に課題名を指定して使用する． 図9に rp_ex1 の内容を示す．

第2行から第5行までの

```
while read f; do; ...; done
```

は，パイプ（または標準入力）から1語読み込んでシェル変数"`$f`" に格納し，"do" から

"done" までの間のコマンドを実行することを，入力がある限り繰り返すことを表す．

```
#!/bin/sh
while read f
do
 cat $HOME/Mail/inbox/$f|rp_ex2 $f $1
done
```

図 9: rp_ex1

本プログラムの場合，"$f" に格納されるのは，電子メールが格納されたファイルのファイル名となる．第4行の

```
cat $HOME/Mail/inbox/$f
```

は，"$f" で示された電子メールの内容を出力することを表す．この出力はパイプによって "rp_ex2 $f $1" に引き渡される．

3.3 rp_ex2

パイプ（または標準入力）から電子メールの内容を入力して，そのメールが課題レポートである場合のみ，その電子メールを格納しているファイル名を出力する．第一引き数に電子メールが格納されているファイル名，第二引き数に課題名を指定して使用する．図 10に rp_ex2 の内容を示す．

```
#!/bin/sh
awk '$1 ~ /Subject:/ {if($2=="'$2'") print "'$1'" }'
```

図 10: rp_ex2

このプログラムは awk を使用している．第2行の

```
$1 ~ /Subject:/
```

は，awk のパターンであり，パイプ（または標準入力）から１行ずつ読み込んで，その行の最初の語である "$1" が "Subject:" と一致した場合に，この式に続く活動部を実行することを表している．同じ第２行の

```
{if($2=="'$2'") print "'$1'" }
```

は，awk のアクションであり，読み込んだ行の２番目の語が第二引き数で与えられた課題名と一致した場合，第一引き数で与えられたファイル名を出力することを表している．

3.4 rp_ex3

パイプ（または標準入力）から課題レポートを格納しているファイル名の一覧を入力して，課題レポートを格納しているファイル名と，そのレポート提出者の組の一覧を出力する．図 11に rp_ex3 の内容を示す．

```
#!/bin/sh
while read f
do
 cat $HOME/Mail/inbox/$f|rp_ex4 $f
done
```

図 11: rp_ex3

第３行の

```
cat $HOME/Mail/inbox/$f
```

は，課題レポートの内容を出力することを表す．この出力には，このレポート提出者の電子メールアドレス等の情報も含まれている．パイプを通じて，この出力は "rp_ex4" に引き渡される．

3.5 rp_ex4

パイプ（または標準入力）から課題レポートの内容を入力して，そのレポートが格納されているファイル名と，レポート提出者の電子メールアドレスを出力する．第一引き数に課題レポートが格納されているファイル名を指定して使用する．図 12に rp_ex4 の内容を示す．

```
#!/bin/sh
awk '$1 ~ /Return-Path:/ { print "'$1'" " " $2 }'
```

図 12: rp_ex4

このプログラムは awk を使用している．第2行のパターン

```
$1 ~ /Return-Path:/
```

は，パイプ（または標準入力）から課題レポートの内容を次々と入力し，レポート提出者の電子メールアドレス（ユーザ名）が記述してある行を見つけた場合に，この式に続く活動部を実行することを表している．なお，電子メールの送り主のアドレスは，"Return-Path:" で始まる行の2番目の語である．同じく第2行のアクション

```
{ print "'$1'" " " $2 }
```

は，第一引き数に与えられた，課題レポートが格納されているファイル名と，そのレポートの提出者のアドレスを出力することを表している．

3.6 rp_ex5

パイプ（または標準入力）から課題レポートを格納しているファイル名と，そのレポート提出者名（ユーザ名）の組の一覧を入力して，課題レポートを

~/report/ 課題名 / 提出者名

に格納し，受けとり通知を電子メールで発行する．第一引き数に，課題レポートが格納されているファイル名を指定して使用する．図13に rp_ex5 の内容を示す．

第2行の

```
while read f g
```

でシェル変数 "$f" に課題レポートが格納されたファイル名が，"$g" にそのレポート提出者のユーザ名が代入される．

第4行の

```
echo $g
```

は現在処理を行なっているレポートの提出者（のユーザ名）を表示する．

```
#!/bin/sh
while read f g
do
 echo $g
 cat $HOME/Mail/inbox/$f > $HOME/report/$1/$g
 rp_ex6 $g $1
 rm $HOME/Mail/inbox/$f
done
```

図 13: rp_ex5

第5行の

```
cat $HOME/Mail/inbox/$f > $HOME/report/$1/$g
```

は，現在メールボックス (~/Mail/inbox) に格納されている課題レポートを，

~/report/ 課題名 / 提出者名

にコピーする．第6行の

```
rp_ex6 $g $1
```

はレポート受けとり通知を電子メールで提出者に送る．第7行の

```
rm $HOME/Mail/inbox/$f
```

はメールボックスに格納されている課題レポートを削除する．

3.7 rp_ex6

レポート受けとり通知を電子メールで提出者に送る．第1引き数に，レポート提出者アドレス，第2引き数に課題名を与えて使用する． 図 14に rp_ex6 の内容を示す．

```
#!/bin/sh
echo "report" $2 "を受けとりました. " > $HOME/report/bin/mtmp
echo $HOME"/report/"$2"/"$1" にありますので確認してください. " >> \
                $HOME/report/bin/mtmp
mail $1 <$HOME/report/bin/mtmp
```

図 14: rp_ex6

第2行の

```
echo "report" $2 "を受けとりました. " > $HOME/report/bin/mtmp
```

は，ファイル ~/report/bin/mtmp に

```
report 課題名 を受けとりました.
```

を書き込む. 第3行から4行にかけての

```
echo $HOME"/report/"$2"/"$1" にありますので確認してください. " >>\
        $HOME/report/bin/mtmp
```

は，先に作られたファイル ~/report/bin/mtmp の末尾に

```
教官ホームデイレクトリ /report/ 課題名 / 提出者名
にありますので確認してください
```

を加える. 第5行の

```
mail $1 <$HOME/report/bin/mtmp
```

は ~/report/bin/mtmp に作成された受けとり通知の内容を，$1 で示されるレポート提出者宛に電子メールで送付する.

4 おわりに

シェルプログラムと awk を組み合わせることによって，短期間で，授業レポート受けとりシステムを作成することができた.

シェルプログラムや awk の他の応用として，授業出席状況採取システム，学生演習状況調査システム，簡単な質問応答システム，簡単な CAI システム，UNIX ワークステーションの管理運営システム等が考えられる．

UNIX には，ファイルやテキストを操作したり，編集したりするコマンドが大量に備わっている．このため，たいていの作業は C 等のプログラミング言語でプログラムを作成する必要はなく，その機能をもったコマンドを探しだすか，またはいくつかのコマンドを，パイプ，リダイレクション，awk やちょっとしたシェルプログラムで組み合わせることによって行なうことができる．

参考文献

[1] S.G. コーチャン /P.H. ウッド著, 玄光男, 荒実訳, *UNIX SHELL* プログラミング,HBJ 出版局,1988.

[2] A.V. エイホ, B.W. カーニハン, P.J. ワインバーガー著 足立高穂訳 プログラミング言語 *AWK*, トッパン, 1989.

[3] R.S. テア著，矢吹道郎監修 田中啓介 / 千種康民 / 野間泉 訳 *UNIX* 基本ツール , マグロウヒル, 1990

[4] 村井純 / 井上尚司 / 砂原秀樹共著 プロフェショナル *UNIX*,ASCII,1986