

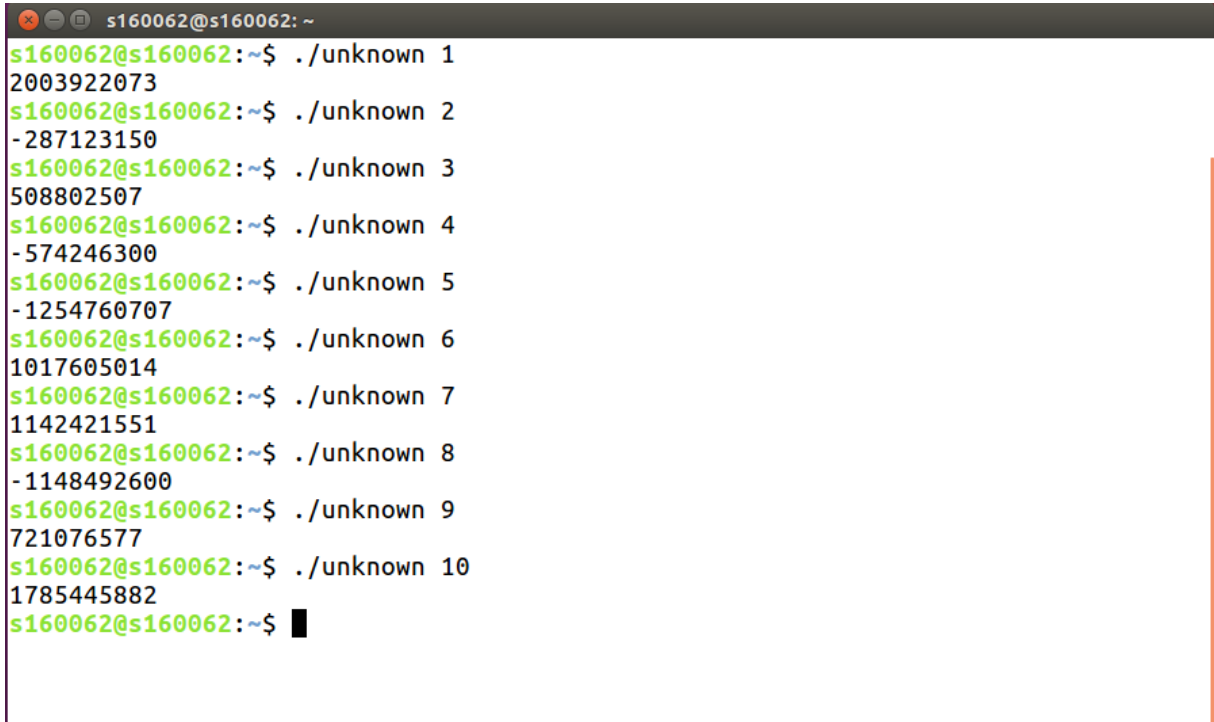
情報工学科 レポート

実験演習記録			判定・指示		
	年月日時	共同作業者			
1	2018/11/13				
2					
3					
4					
レポート提出記録					
	提出年月日	期限年月日			
初	2018/11/27	2018/11/27			
再					
科目名		テーマ担当教員	学年	学期	単位
実験1		山田先生	2	後期	2
テーマ番号 テーマ名		学籍番号 氏 名			
シェルスクリプト		16268062 Salic Ertugrul			

1. 課題 1

課題 1-1

実行結果を図 1 に示す．結果は入力する数字によって正か負の数になっている．



```
s160062@s160062: ~  
s160062@s160062:~$ ./unknown 1  
2003922073  
s160062@s160062:~$ ./unknown 2  
-287123150  
s160062@s160062:~$ ./unknown 3  
508802507  
s160062@s160062:~$ ./unknown 4  
-574246300  
s160062@s160062:~$ ./unknown 5  
-1254760707  
s160062@s160062:~$ ./unknown 6  
1017605014  
s160062@s160062:~$ ./unknown 7  
1142421551  
s160062@s160062:~$ ./unknown 8  
-1148492600  
s160062@s160062:~$ ./unknown 9  
721076577  
s160062@s160062:~$ ./unknown 10  
1785445882  
s160062@s160062:~$ █
```

図 1 課題 1-1 unknown に 1～10 の入力を与えた結果

課題 1-2

作成したスクリプトを次に示す．

課題 1-2 のソースコード

```
#!/bin/bash  
for i in {1..500}  
do  
./unknown $i >> unknown.log  
done
```

for 文で i 変数に 1 から 500 までの数を代入しながら, i を unknown に入力として与えていく．そして unknown.log というファイルに出力を行う．プログラムの実行結果 (unknown.log) を本レポートに添付して提出する．

課題 1-3

以下に制作したプログラムのソースを示す.

課題 1-3 のソースコード

```
#!/bin/bash

saidaichi=0
saishouchi=0
for i in {1..500}
do
    a=$(./unknown $i)
    echo "$a"
    if [ "$a" -gt "$saidaichi" ]
    then
        saidaichi=$a
    fi
    if [ "$a" -lt "$saishouchi" ]
    then
        saishouchi=$a
    fi

    echo "入力 : $i"
    echo "今までの最大値 : $saidaichi"
    echo "今までの最小値 : $saishouchi"

done
```

まず, 最大値を指す saidaichi と最小値を指す saishouchi 変数を 0 で初期化しておく. 次に, for ループに入って unknown を実行し, その結果を \$a に代入する. \$a が \$saidaichi より大きければ \$saidaichi を \$a で更新する. 同じように a が \$saishouchi より小さければ \$saishouchi を \$a で更新する. そして入力が 500 にまで入力, その時までの最大値, 最小値をそれぞれ出力させる. それでプログラムが終了する.

その結果の一部を図 2 に示す.

```

s160062@s160062:~$ ./kadai1_3.sh
入力：1
今までの最大値：2003922073
今までの最小値：0
入力：2
今までの最大値：2003922073
今までの最小値：-287123150
入力：3
今までの最大値：2003922073
今までの最小値：-287123150
入力：4
今までの最大値：2003922073
今までの最小値：-574246300
入力：5
今までの最大値：2003922073
今までの最小値：-1254760707
入力：6
今までの最大値：2003922073
今までの最小値：-1254760707
入力：7
今までの最大値：2003922073
今までの最小値：-1254760707
入力：8
今までの最大値：2003922073
今までの最小値：-1254760707
入力：9
今までの最大値：2003922073
今までの最小値：-1254760707
入力：10
今までの最大値：2003922073
今までの最小値：-1254760707
入力：11
今までの最大値：2003922073
今までの最小値：-1847686509
入力：12
今までの最大値：2035210028
今までの最小値：-1847686509
入力：13
今までの最大値：2035210028
今までの最小値：-1847686509

```

図 2 課題 1-2 unknown に 1～500 の入力を与え，最大値と最小値を求めた結果

2. 課題 2

課題 2-1

以下に今回制作したプログラムのソースを示す。

課題 2-1 のソースコード

```
ls -hlS | awk '{print $9}----->"$5}'|sed -n 2,4p
```

ls -hlS: ls はファイルやディレクトリの情報を表示するコマンドである。-l オプションでリスト表示ができる。そして-S オプションでファイルをサイズ順（降順）にソートして表示する。-h はサイズの表示を k, M, G のようなシンボルで分かりやすくする。今回はそれらの組み合わせを用いた。

awk '{print \$9}----->"\$5}': awk コマンドで 9 番目の列である「ファイル名」と 5 番目の列である「サイズ」を出力する。今回はその間に読みやすくなるよう“----->”も入れた。

sed -n 2,4p : 2 行目から 4 行目までを表示する。head -4 を用いても良かったが、そうだと 1 行目が課題と関係のない出力となってしまうので、今回は sed コマンドを用いた。

課題 2-1 のコードの実行結果を図 3 に示す。

```
s160062@s160062:~$ ls -hlS | awk '{print $9}----->"$5}' | sed -n 2,4p
linux-4.19.1.tar.xz----->99M
cikti.txt----->31K
unknown.log----->11K
s160062@s160062:~$
s160062@s160062:~$ █
```

図 3 課題 2-1 ファイルサイズの大きい上位 3 つのファイルの出力

課題 2-2

課題 2-2 において作成したスクリプトを次に示す。

課題 2-2 のソースコード

```
#!/bin/bash
(
    grep -lr "linus" linux-4.19.1/ >> temp.txt

    echo ===== TEXT FILES =====
    grep "¥(¥.txt¥)$" temp.txt

    echo ===== C FILES =====
    grep "¥(¥.c¥)$" temp.txt

    echo ===== H FILES =====
    grep "¥(¥.h¥)$" temp.txt
) >> linus.log
rm -f temp.txt
```

```
grep -lr "linus" linux-4.19.1/ >> temp.txt
```

上のコードでは `grep` の `-l` (エル) と `-r` オプションを用いた。「`-l` (エル)」オプションは検索した結果にファイル名のみを表示するのに用いる。「`-r`」はディレクトリ内の検索を行うのに必要なオプションである。上のコマンドによって、「linux-4.19.1/」ディレクトリ内に存在して、「linus」という文字列を含むようなファイルのファイル名を `temp.txt` というテキストファイルに出力できる。

```
grep "¥(¥.txt¥)$" temp.txt ;;; grep "¥(¥.c¥)$" temp.txt ;;; grep "¥(¥.h¥)$" temp.txt
```

上の3つのコマンドも同じような処理を行う。`temp.txt` に保存したデータの中から、ファイル名の最後に“.txt”、“.c”、“.h”のあるファイルを見つけて出力する。そのためには正規表現を使用する。

```
>> linus.log
```

上記のコードをカッコ () で一つにまとめて、その結果を `linus.log` というファイルに保存する。

```
rm -f temp.txt
```

一時的に使用した `temp.txt` というファイルを削除する。エラーメッセージや警告などが出ないように `-f(force)` オプションを用いる。

課題 2-2 で作成したプログラムの実行結果（すなわち `linus.log`）を本レポートに添付して提出する。

3. 課題 3

課題 3-1

課題 3-1 では与えられたデータの最大値と最小値を求めた。`weatherF.dat` にあるデータを、各列の降順で並べ替え、その中から、先頭行と最終行を、つまり最大値と最小値を取り出せば良い。作成したプログラムを次に示す。（プログラムをそのままコピーするとエラーが出てしまう。各行の先頭にあるスペースを消してから実行することに注意してもらいたい。）

次に示すプログラムでは `sort` コマンドを用いて各列の降順されたデータから先頭行と最終行を取り出す。`cut` コマンドで取り出した行の必要な列を出力させる。

課題 3-1 において作成したシェルスクリプトのプログラムの実行結果を図 4 に示す。

```
#!/bin/bash
#line という配列に各列の名前を代入しておく
for i in {1..9}
do
    line[$i]=`cut -f $i weatherF.dat | head -1`
done

for i in {2..9}
do
    echo ${line[$i]}

    cat weatherF.dat |¥
    sed '$d' | #データだけを取り出す（スペースである最終行を消す）
    tail -n +2 | #データだけを取り出す（文字列からなる先頭行を消す）
    sort -nrk $i | #降順に並べ替える
    (head -1; tail -n 1) | #先頭行と最終行を取り出す
    cut -f 1,$i #特定の列を取り出す
    echo
done
```

```
s160062@s160062:~$ ./kadai3_1.sh
Ave. temperature (C)
2017/7/17      30.1
2017/1/15     -0.3

Highest tempareture (C)
2017/8/9       37.6
2017/1/20       3.1

Lowest tempareture (C)
2017/8/8       26.8
2017/1/15     -6.5

Precipitation (mm)
2017/10/22    158.5
2017/10/10       0

Day length (hours)
2017/7/9      12.9
2017/10/11       0

Ave. wind(m/s)
2017/4/7       4.5
2017/1/7       0.7

Maximum wind (m/s)
2017/9/18     10.9
2017/10/21      1.6

Maximum instantaneous wind (m/s)
2017/9/18     24.3
2017/10/28      3.7
```

図 4 課題 3-1 weatherF.dat の各列の最大値と最小値

課題 3-2

課題 3-2 では、プログラム言語としてシェルスクリプトを用い、weatherF.dat の一日の平均気温、降水量、日照時間の月別の平均値を出力するスクリプトを作成する。そのためには awk コマンドを用いればよい。まず、課題 3-1 でしたように、文字列やスペースを消し、数字（データ）のみを取り出す。for ループで 1 月から 10 月までの各月のデータを grep コマンドで取る。その際に、一時記憶ファイルとして temp3_2.txt を作るが、プログラムの最後に rm -f コマンドで削除する。そして awk コマンドで各月の 2 列目（一日の平均気温）、5 列目（降水量）、6 列目（日照時間）の平均を求める。次のコードを例として説明をする。

```
awk '{sum2 += $2}END{printf sum2 / NR}'
```

上のコマンドでは 2 列目の全ての要素を sum2 変数に足してゆき、最後に sum2 を行数 (NR) で割ることによって平均値を求める。そして求めた値を出力する。

課題 3-2 において作成したプログラムの実行結果を図 5 に示す。

```
s160062@s160062:~$ ./kadai3_2.sh
1月の平均値
Ave. temperature (C): 4.83226
Precipitation (mm): 0.774194
Day length (hours): 7.5

2月の平均値
Ave. temperature (C): 6.09643
Precipitation (mm): 0.482143
Day length (hours): 7.43571

3月の平均値
Ave. temperature (C): 7.72581
Precipitation (mm): 2.93548
Day length (hours): 5.79032

4月の平均値
Ave. temperature (C): 14.04
Precipitation (mm): 3.31667
Day length (hours): 6.53

5月の平均値
Ave. temperature (C): 19.5419
Precipitation (mm): 1.72581
Day length (hours): 6.79677

6月の平均値
Ave. temperature (C): 21.7133
Precipitation (mm): 2.88333
Day length (hours): 5.28

7月の平均値
Ave. temperature (C): 27.0935
Precipitation (mm): 5.40323
Day length (hours): 6.60968

8月の平均値
Ave. temperature (C): 26.1452
Precipitation (mm): 3.77419
Day length (hours): 2.83548

9月の平均値
Ave. temperature (C): 22.29
Precipitation (mm): 6.06667
Day length (hours): 4.38

10月の平均値
Ave. temperature (C): 16.2839
Precipitation (mm): 18.7097
Day length (hours): 3.13871
```

図 5 課題 3-2 weatherF.dat の各月の 2・5・6 列目の平均値

課題 3-2 のソースコード

```
#!/bin/bash
for i in {1..9}
do
    line[$i]=`cut -f $i weatherF.dat | head -1`
done
for i in {1..10}
do
    cat weatherF.dat |
    sed '$d' |
    tail -n +2 |
    grep "2017/${i}/" > tmp3_2.txt
    echo "$i月の平均値"
    echo -n "${line[2]}: " #一日の平均温
    cat tmp3_2.txt|
    awk '{sum2 += $2}END{printf sum2 / NR}'
    echo
    echo -n "${line[5]}: " #降水量
    cat tmp3_2.txt|
    awk '{sum5 += $5}END{printf sum5 / NR}'
    echo
    echo -n "${line[6]}: " #日照時間
    cat tmp3_2.txt|
    awk '{sum6 += $6}END{printf sum6 / NR}'
    echo
    echo
done
rm -f tmp3_2.txt
```

課題 3-3

課題 3-3 ではプログラム言語としてシェルスクリプトを用いた。今回は都市名も出力するので、最初に、各都市データの各行の先頭にその都市名を入れることにした。行先頭に都市名という列を入れたデータを、一時記憶ファイルとして使用する tmp3_3.txt に保存する。ここまでの動作をそれを次のコマンドで行う。

```

(for i in {W,F,K}
do
    sed "s/^/$i\t/g" weather$i.dat |
    sed '$d' |
    tail -n +2
done )>tmp3_3.txt

```

上のコードでは weatherW.dat, weatherF.dat, weatherK.dat のファイルからデータを読み取って、そのデータの最初に W,F,K を各行に列として加える。また、文字列とスペースからなる先頭行と最終行を消す。それによって数字のみからなるデータが手に入る。残ったデータを一時記憶ファイルに書き込む。

次に、array という配列を定義し、その配列に awk コマンドで求めた平均値を格納する。今回はデータの最初に都市名という列を加えたので、今回使用する列は 3・4・5 列目である。今度使うのが簡単のため、array の 3・4・5 番目の要素に値を入れることにした。そのコードを次に示す。

```

declare -a array
array[3]='awk '{sum3 += $3}END{print sum3 / NR}' tmp3_3.txt`
array[4]='awk '{sum4 += $4}END{print sum4 / NR}' tmp3_3.txt`
array[5]='awk '{sum5 += $5}END{print sum5 / NR}' tmp3_3.txt`

```

最後に、課題 3-1 の設計を用いて、最大値と最小値を求める。すなわち、一時記憶ファイルのデータを降順でソートし、その先頭行（最大値）と最終行（最小値）を取り出す。コードは次の通りである。なお、line 配列は列名を指すものである。そして平均値を格納した配列とともに出力をする。

```

for i in {3,4,5}
do
    echo ${line[$i]}
    cat tmp3_3.txt |
    sort -nrk $i |
    (head -1; tail -n 1) |
    awk -v I=$i '{print $I "\t"$2 " in " $1 " city"}' |
    sed '1s/^/最大値:/' | sed '2s/^/最小値:/'
    echo "平均値:${array[$i]}"
    echo
done

```

課題 3-3 で作成したプログラム全体を次に示す。

課題 3-3 のソースコード

```
#!/bin/bash
for i in {1..9}
do
    line[$i+1]=`cut -f $i weatherF.dat | head -1`
done

(for i in {W,F,K}
do
    sed "s/^/$i¥t/g" weather$i.dat |
    sed '$d' |
    tail -n +2
done )>tmp3_3.txt

declare -a array
array[3]=`awk '{sum3 += $3}END{print sum3 / NR}' tmp3_3.txt`
array[4]=`awk '{sum4 += $4}END{print sum4 / NR}' tmp3_3.txt`
array[5]=`awk '{sum5 += $5}END{print sum5 / NR}' tmp3_3.txt`

for i in {3,4,5}
do
    echo ${line[$i]}
    cat tmp3_3.txt |
    sort -nrk $i |
    (head -1; tail -n 1) |
    awk -v I=$i '{print $I "¥t"$2 " in " $1" city"}' |
    sed '1s/^/最大値:/' | sed '2s/^/最小値:/'
    echo "平均値:${array[$i]}"
    echo
done

rm tmp3_3.txt
```

課題 3-3 のプログラムの実行結果を図 6 に示す.

```
@s160062: ~/3-3
s160062@s160062:~/3-3$ ./kadai3_3.sh
Ave. temperature (C)
最大值:31.2      2017/8/8 in K city
最小値:-10.7    2017/1/11 in W city
平均値:15.0069

Highest tempareture (C)
最大值:37.6      2017/8/9 in F city
最小値:-8.9      2017/1/11 in W city
平均値:19.0255

Lowest tempareture (C)
最大值:28.6      2017/8/9 in K city
最小値:-12       2017/1/11 in W city
平均値:11.5105
```

図 6 課題 3-3 プログラムの実行結果

4. 課題 4

課題 4 では、プログラミング言語として python を用いる。

課題 4-1

次に作成したプログラムのソースを示す。

```
i=0
live=0
dead=0
total=0
ll=3
t_rack=0
n_rack=0
dict_rack={0:0}

for i in range (1,101):
    s = "rack"+str(i)+".log"
    f = open(s,'r')
    jisyo=f.readline()
    while jisyo:
```

```

        jisyo = f.readline().split()
        if len(jisyo) == 0:
            continue
        if jisyo[1]=='OK':
            live+=1
        if jisyo[1]=='DOWN':
            dead+=1
        t_rack+=int(jisyo[2])
        n_rack+=1
    dict Rack.update([(s, t_rack/n_rack)])
dict_sort=sorted(dict Rack.items(), key=lambda x:x[1],reverse=True)
total=int(live+dead)

print('現在稼働中のサーバーは%d/%d = %f%%' % (live, total, 100*live/total))
print('\n1 位:')
print(dict_sort[0])
print('\n2 位')
print(dict_sort[1])
print('\n3 位')
print(dict_sort[2])

```

次の図 7 に実行結果を示す。

```

In [27]: runfile('C:/Users/Katre Jp/log/kadai4_1.py', wdir='C:/Users/Katre Jp/log')
現在稼働中のサーバーは25312/25900 = 97.729730%

1位:
('rack25.log', 2517.816)

2位
('rack26.log', 2516.1210447761196)

3位
('rack7.log', 2515.212)

```

図 7 課題 4-1 の実行結果

課題 4-2

以下に今回作成したプログラムのソースを示す。

```

i=0
mainte=0
disk=0

```

```

power=0
cpu=0
memory=0
unknown=0
for i in range (1,101):
    s = "rack"+str(i)+".log"
    f = open(s,'r')
    jisyo=f.readline()
    while jisyo:
        jisyo = f.readline().split()
        if len(jisyo) == 0:
            continue
        if jisyo[3]=='MAINTENANCE':
            mainte+=1
        if jisyo[3]=='DISK':
            disk+=1
        if jisyo[3]=='POWER':
            power+=1
        if jisyo[3]=='CPU':
            cpu+=1
        if jisyo[3]=='MEMORY':
            memory+=1
        if jisyo[3]=='UNKNOWN':
            unknown+=1

print(' 停止の原因はメンテナンス%d 件\nディスク故障%d 件\n電源生涯%d 件\nCPU のエラー%d 件\nメモリエラ
ー%d 件\n原因不明%d 件' % (mainte, disk,power, cpu, memory, unknown))

```

プログラムの実行結果を図 8 に示す.

```

In [28]: runfile('C:/Users/Katre Jp/log/kadai4_2.py', wdir='C:/Users/Katre Jp/log')
停止の原因はメンテナンス24239件
ディスク故障214件
電源生涯414件
CPUのエラー257件
メモリエラー264件
原因不明512件

```

図 7

課題 4-3

提出期限が迫ってしまい，プログラムの作成と実装ができなかった．

(2 週間が 1 つの実験に足りるはずだったのですが，今回は試験などとも重なってしまい，間に合わせられませんでした．大変申し訳ございません．ですが，面白いので，レポートの提出の後に本課題と課題 4-4 にも取り組んでみます．きちんとしたレポートが書けたら，フィードバックを希望したいと思っていましたが，今回のレポートは丁寧に書けていないので，間違いなどがたくさんあることで，フィードバックの希望をやめます．本当にありがとうございます.)