Systems Programming

09 - Shells Scripting

redirection

new file

cth: `ls > isi-dari-ls`

>

append file

cth: `ls -al >> isi-dari-ls`

>>

stdin dari file

cth: `sort < isi-dari-ls`

< <<

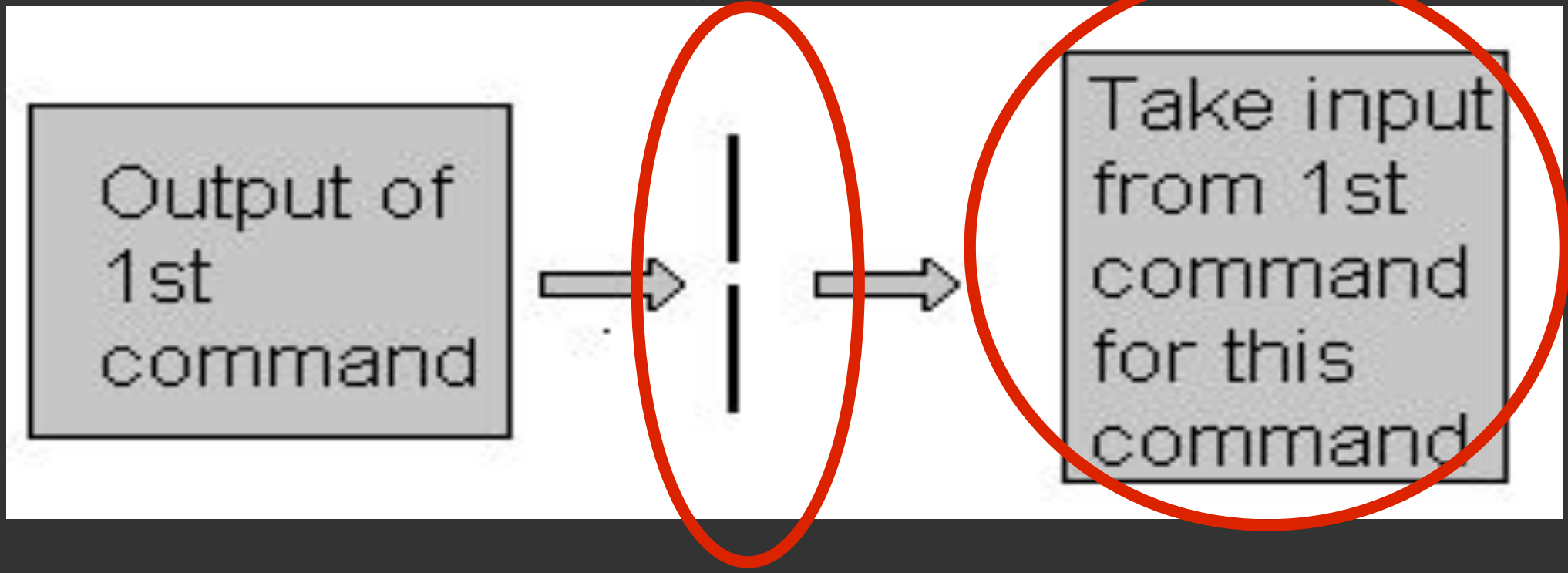
pipe

a way to connect the output of one program to the input of another program without any temporary file

filter

a filter performs some kind of process on the input and gives output

pipe & filter


Output of 1st command
Take input from 1st command for this command

simple_read.c

```
#include <unistd.h>
#include <stdlib.h>

int main(){
     char buffer[128];
     int nread;

     nread = read(0, buffer, 128);
     if (nread == -1)
          write(2, "Read error\n", 11);
     if ((write(1,buffer,nread)) != nread)
          write(2, "Write error\n",12);
     exit(0);

}
```

simple_writeX.c

```
#include <unistd.h>
#include <stdlib.h>

int main()
{
    write(1, "This is Standard Output\n", 24);
    write(2, " This is Standard Error\n",  23);
    exit(0);
}
```

pipe & filter

ex:

logs file (logs.txt) with 10k lines

retrieve line 423 to 3221

```
# head -n 3221 < logs.txt | tail -n +423
```

operation

```
for { variable name } in { list }
do
    execute one for each item in
    the list until the list is not
    finished (and repeat all
    statement between do and done)
done
```

syntax #1 :: for

operation

```
for (( expr1; expr2; expr3 ))
do
     .....
     ...
     repeat all statements between do
    and done until expr2 is TRUE
done
```

syntax #2 :: for

operation

```
#!/bin/bash
i=1
for day in Mon Tue Wed Thu Fri
do
 echo "Weekday $((i++)) : $day"
done
```

operation

```
while [ condition ]
do
    command1
    command2
    command3
    ....
done
```

syntax :: while

operation

```
#!/bin/bash
n=1
while [ $n -lt 5 ]
do
        echo "Welcome $n times."
        n=$(( n+1 ))
done
```

operation

```
#!/bin/bash
n=1
while (( $n <= 5 ))
do
        echo "Welcome $n times."
        n=$(( n+1 ))
done
```

additional tools

Sed: stream editor

sed : NON-interactive editor

- **sed 's/funtion/function/g' < mainx.c > main.c**
- **sed -e '4d' -e '2d' hapus-b4-b3.txt**
- **sed -e '1d' -e '$d' -e '/^$/d' hapus-b1-kosong.txt**
- **sed 's/\([^:]*\).*/\1/' /etc/passwd**
- **sed 's/\(^\|[^0-9.]\)\([0-9]\+\)\([0-9]\{3\}\)/\1\2,\3/g' numbers**

REGEX (REGular EXpression)

- How to find a word ‘kambing’ in a file?
- How to find an HTML TAG in a page (.html)?
- How to filter an email address in a logs file?
- The answer : REGEX!
- You have learned this before : TBA/Automata Theory & Languages

Put it into practice

"I scream, you scream, we all scream for ice cream."

- How to find the first 'scream'?
- How to find 'cream', not 'scream' nor 'Cream'?
- etc...

Things to remember

- There are a lot of Regex dialect, some depend on the language itself (Perl, Php, .Net, Java)
- Historically, the regex processing is per-LINE
- Find your own guide, every man to his taste
- Practice makes perfect!
- DO NOT REGEX CRAMMING

Examples

- Email Address

  \b[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,4}\b

- <TAG HTML> ZCZC BLAH BLAH</TAG>

  <TAG\b[^>]*>(.*?)</TAG>

- TAG HTML 2

  <([A-Z][A-Z0-9]*)\b[^>]*>(.*?)</\1>

- CASELESS

  \1

Special Characters

[\ ^ $. | ? * + ( )

- Use 'escape' “\”
  - from: “ 2 + 2 = 4 “
  - to: “ 2 \+ 2 = 4”
- Single digit numbers: \d
  - 0 1 2 3 4 5 6 7 8 9
- \w = A-za-z0-9_
- \s = whitespace

Special Characters 2

- \t = tab
- \r = <CR> -carriage return
- \n = Line Feed
- \xFF = HEX
- \uFFFF = Unicode
- \b = word boundary
- \B = does not match \b

Common Characters

- Pal[au] = Pala or Palu
- [0123456789] = 0 or 1 or 2 or ...
- [0-9] = [0123456789]
- [A-Z] = A or B or ...
- [A-Za-z] = one alphabet (capital or not)
- Ak[^a] = Not Ak and not Aka.

Dot

- “.” for any characters
- **\d\d.\d\d.\d\d\d\d**
  - Both 10/10/2010 and 10a10a2010 are possible
- **\d\d[/- .]\d\d[/- .]\d\d\d\d**
  - Both 10/10/2010 and 10/10.2010
  - As well as 99/99/9999
- **[0-3]\d[/- .][0-1]\d[/- .]\d\d\d\d**
  - 31/12/2010 and 33/13/9999

The question mark (?)

- colou?r = color and colour
- Nov(ember)? = Nov and November
- Feb(ruary)? 23(rd)?
  - Feb 23
  - Feb 23rd
  - February 23
  - February 23rd

^ & $

- ^a : front of -> **a**rdhi
- i$: on the back -> ardh**i**
- \d+ sdsdjh345kjkjk
- ^\d+$ 345

```
/usr/share/common-licenses
```

grep "^GNU" GPL-3

grep "and$" GPL-3

grep "[^c]ode" GPL-3

grep "t[wo]o" GPL-3

Curly brackets { }

- \b[1-9][0-9]{3}\b
  - 1000 - 9999
- \b[1-9][0-9]{2,4}\b
  - 100 - 99999

- { } : The number of intended occurrence of the characters

AWK

- Kind of ‘reporting tools’
- Filtering file
- Per-line processing :

pattern { action }

```
BEGIN { print "START" }
      { print "PROCESSING" }
END   { print "FINISH" }
```

AWK in action

```
$ last
rms46    pts/0        jembatan.cs.ui.a Sun Sep 25 22:16   still logged in
julia.ed pts/0        kawung.cs.ui.ac. Sun Sep 25 18:35 - 18:35  (00:00)
rizki.ma pts/0        kawung.cs.ui.ac. Sun Sep 25 15:54 - 16:25  (00:30)
rizki.ma pts/0        kawung.cs.ui.ac. Sun Sep 25 15:52 - 15:54  (00:01)
rizki.ma pts/0        kawung.cs.ui.ac. Sun Sep 25 15:29 - 15:51  (00:22)
rizki.ma pts/0        kawung.cs.ui.ac. Sun Sep 25 15:28 - 15:28  (00:00)
rizki.ma pts/0        kawung.cs.ui.ac. Sun Sep 25 14:50 - 15:27  (00:37)
adrianto pts/0        kawung.cs.ui.ac. Sat Sep 24 23:19 - 02:34  (03:15)
adrianto pts/0        kawung.cs.ui.ac. Sat Sep 24 23:14 - 23:16  (00:01)
adrianto pts/0        kawung.cs.ui.ac. Sat Sep 24 22:50 - 23:01  (00:11)
adrianto pts/0        kawung.cs.ui.ac. Sat Sep 24 22:46 - 22:47  (00:00)
andrea.b pts/1        kawung.cs.ui.ac. Sat Sep 24 20:12 - 22:25  (02:12)

$ last | awk '{print $1}'
rms46
julia.ed
rizki.ma
rizki.ma
rizki.ma
rizki.ma
rizki.ma
adrianto
adrianto
adrianto
adrianto
andrea.b
```

Filtering and piping

```
$ last | awk '{print $1}' | sort -u

abdel.ja
ade.rahm
adilla.w
adrian.a
adrianto
aji.prad
anandra.
andika.w
andrea.b
ardhi.pu
ardhiwib
arif.fai
```

Exercise

Have fun!

Create your script

1) create the script file

```
shell> vi suatu-script.sh
```

2) declare that file “suatu-script.sh” can be executed in shell

   Hint: read ***man chmod*** **!!!**

```
shell> chmod +x suatu-script.sh
```

suatu-script.sh content

```
#!/bin/bash

echo "Hello world"
```

Execute the script

executing

```
shell> ./suatu-script.sh
```

or

```
shell> bash suatu-script.sh
```

Exercise #1

How many total bytes recorded from file ***apache.txt*** *(20000 lines) between line #335 to #14105 ?*
Put that number to file named ***total.txt***

Download apache.txt file by

```
wget
https://projects.ui.ac.id/attac
hments/7271/apache.txt
--no-check-certificate
```

Download the apache.txt

wget --no-check-certificate
https://projects.ui.ac.id/attachments/7271/apache.txt

```
#!/bin/bash

head -n 14105 < apache.txt | tail -n +335 >
tmp.txt

awk 'BEGIN{
total = 0;
}
{
sub(/^[^"]*"[^"]*" [^ ]* /,"");
total += $1;
}
END {
printf("%d\n", total);}' tmp.txt > total.txt
```

Exercise #2

Retrieve unique IP from data in exercise #1, then, put those into file unik.txt

```
#!/bin/bash

head -n 14105 < apache.txt | tail -n +335
>tmp2.txt

awk '{
printf("%s\n", $1);
}' tmp2.txt | sort -u | wc -l > unik.txt
```

Exercise #3

Find which files that are most downloaded from the previous data. Also, include its absolute path. If there are two or more files that have the same amount, just pick one of them. Put your findings in ***sering.txt***

```
#!/bin/bash

head -n 14105 < apache.txt | tail -n +335 >
tmp3.txt

grep GET tmp3.txt | awk 'BEGIN{
fname = "";
jumlahDL = 0;
}
{
sub(/^[^"]*"GET /,"");
download[$1]++
}
END {
for(DL in download){
if(download[DL]>jumlahDL){
jumlahDL = download[DL];
fname = DL; }}
printf("%s\n", namaBerkas);}' > sering.txt
```

Exercise #4

Create a report containing the top downloader ***in a particular minute*** *from data in exercise #2. Only consider total bytes on* ***HTTP/1.1 with Response 200*** *as its parameter. Also, note the IP address. Put the* ***time information (hh:mm)*** *<space>* ***total byte downloaded for all IP*** *<space>* ***IP of top downloader*** *to a file report.txt.*

Example :

09:54 274432 152.118.212.43
09:55 1992243 152.118.80.2
09:56 45644 152.118.21.55

Hint for exercise #4

1. *Total Byte per minute is a total byte transferred in a particular* ***hour::minute***
2. *HTTP/1.1 Response 200 is string* ***HTTP/1.1" 200***
3. *Tips: use Array with associative index* ***"Hour:minute IP"***

```
grep "HTTP/1.1\" 200" proses.txt | awk 'BEGIN{
   IPMaks= "";
   jumlahDL = 0;
} {
   sub(/-[^:]*:/,"");
   sub(/:[0-9][0-9] [^"]*"[^"]*" [^ ]*/,"");
   total[$2] += $3;
   IP[$2, $1] += $3;
} END {
   for(jam in total) {
      IPMaks = "";
      jumlahDL = 0;
      for(pengguna in IP) {
         if((jam == substr(pengguna,1,5))&&(IP[pengguna]
> jumlahDL)) {
            IPMaks = substr(pengguna,6);
            jumlahDL = IP[pengguna];
         }
      }
      printf("%s %d %s\n", jam, total[jam],IPMaks);
   }
}'| sort -k1 -n > laporan.txt
```

Exercise #5

Write a script called 'detect.sh' to detect disconnected/connected network cable.
It should behave like as follows :

1. *If the cable is disconnected, it should show **'DISCONNECTED.'** Otherwise (if it is connected), show **'CONNECTED'**.*
2. *Checking interval is **1 second.***
3. *The status is only shown if and only if **there is an actual state change** in the network cable.*

Hint for Exercise #5

1. *use* ***while true BASH, if-else-fi***
2. *useful commands :* ***dmesg, cut, sleep***
3. *find* ***“link up”/“link-down”*** *phrase from kernel message*

```
#!/bin/bash

STATUS="awal"

while true;
do
        LAST=`dmesg | grep e1000 | tail -n 1 | cut -d " "
-f 7`
        if [ $STATUS != $LAST ]
        then
                STATUS=$LAST
                if [ $STATUS == "Down" ]
                then
                        echo "DISCONNECTED"

                else
                        echo "CONNECTED"
                fi
        fi
        sleep 1
done
```

discussion time

Ilustrasi Pipa dan Filter (pada Rokok)