# Standard I/O in Linux, pipes, redirection, regular expressions, text processing

Nikita Neveditsin, SMU, 2019

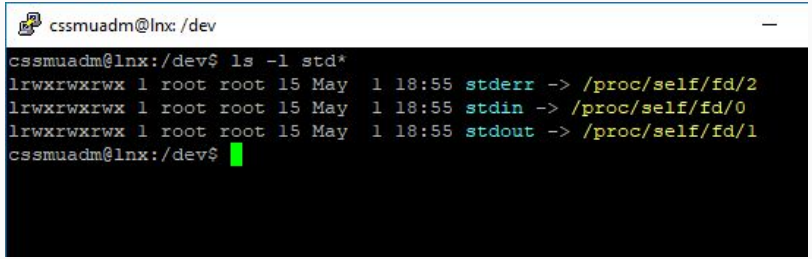
nikita.neveditsin@smu.ca

# Standard Input and Output

In Linux all processes are given 3 streams for **input** and **output** of data:

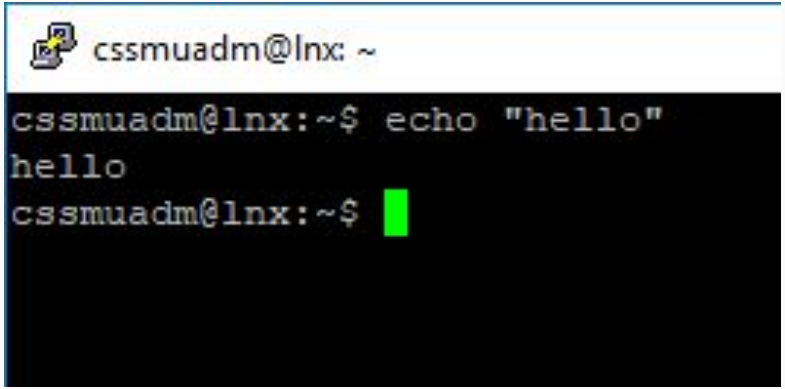
- Standard Input (STDIN) - file descriptor 0
- Standard Output (STDOUT) - file descriptor 1
- Standard Error (STDERR) - file descriptor 2

STDIN “connected” to keyboard by default, STDOUT connected to the console/terminal as well as STDERR



# Standard Input and Output

**echo** command displays text using standard output



# Standard Input and Output

Very simple **di** command:

- Reads from Standard Input
- Outputs it double times (*:*) to STDOUT
- Outputs message “STDERR OUTPUT: *” to STDERR

cssmuadm@lnx: ~

```
GNU nano 2.5.3            File: di.c

#include <stdio.h>
int main()
{
   char str[1024] = {0};
   fscanf(stdin, "%s", (char*)&str);
   fprintf(stdout, "%s:%s\n", str, str);
   fprintf(stderr, "STDERR OUTPUT: '%s'\n", str);
}
```

cssmuadm@lnx: ~

```
cssmuadm@lnx:~$ di
123
123:123
STDERR OUTPUT: '123'
cssmuadm@lnx:~$
```

# Standard Input and Output: Pipes

Pipe (|) sends output of one program to input of another program. It's one of the most popular methods of interprocess communication in Linux

- Programs can work in parallel
- Pipes have limited size and allocated in *pipefs* (in memory file system)
- If pipe is full, write blocks until pipe has enough space to write, if pipe is empty, read blocks until there is some data to read from pipe

cssmuadm@lnx: ~

```
cssmuadm@lnx:~$ echo "1" | di
1:1
STDERR OUTPUT: '1'
cssmuadm@lnx:~$ echo "1" | di | di
STDERR OUTPUT: '1'
1:1:1:1
STDERR OUTPUT: '1:1'
cssmuadm@lnx:~$ echo "1" | di | di | di
STDERR OUTPUT: '1'
STDERR OUTPUT: '1:1'
1:1:1:1:1:1:1:1
STDERR OUTPUT: '1:1:1:1'
cssmuadm@lnx:~$ echo "1" | di | di | di
```

cssmuadm@lnx: /proc/5274/fd

```
-r--r--r-- 1 cssmuadm cssmuadm 0 May  9 16:09 status
-r-------- 1 cssmuadm cssmuadm 0 May  9 16:22 syscall
dr-xr-xr-x 3 cssmuadm cssmuadm 0 May  9 16:22 task
-r--r--r-- 1 cssmuadm cssmuadm 0 May  9 16:22 timers
-rw-r--r-- 1 cssmuadm cssmuadm 0 May  9 16:22 uid_map
-r--r--r-- 1 cssmuadm cssmuadm 0 May  9 16:22 wchan
cssmuadm@lnx:/proc/5274$ cd fd
cssmuadm@lnx:/proc/5274/fd$ ls -l
total 0
lrwx------ 1 cssmuadm cssmuadm 64 May  9 16:22 0 -> /dev/pts/0
l-wx------ 1 cssmuadm cssmuadm 64 May  9 16:22 1 -> pipe:[201843]
lrwx------ 1 cssmuadm cssmuadm 64 May  9 16:20 2 -> /dev/pts/0
cssmuadm@lnx:/proc/5274/fd$
```

STDOUT redirected to pipe

# Standard Input and Output: Redirection

Redirect output (>) redirects output (STDOUT) to **file**. The same as 1> as 1 stays for stdout file descriptor

cssmuadm@lnx: ~

```
cssmuadm@lnx:~$ ls -l
total 2444
-rw-rw-r-- 1 cssmuadm cssmuadm      12 May  9 14:35 1
-rwxrwxr-x 1 cssmuadm cssmuadm    8856 May  9 14:35 di
-rw-rw-r-- 1 cssmuadm cssmuadm     187 May  9 14:35 di.c
-rw-rw-r-- 1 cssmuadm cssmuadm      35 May  9 18:02 out.txt
drwxrwxr-x 3 cssmuadm cssmuadm    4096 May  7 14:11 tmp
-rw-r--r-- 1 cssmuadm cssmuadm 2473400 May  7 14:12 world192.txt
cssmuadm@lnx:~$ ls -l > out.txt
cssmuadm@lnx:~$ cat out.txt
total 2440
-rw-rw-r-- 1 cssmuadm cssmuadm      12 May  9 14:35 1
-rwxrwxr-x 1 cssmuadm cssmuadm    8856 May  9 14:35 di
-rw-rw-r-- 1 cssmuadm cssmuadm     187 May  9 14:35 di.c
-rw-rw-r-- 1 cssmuadm cssmuadm       0 May  9 18:02 out.txt
drwxrwxr-x 3 cssmuadm cssmuadm    4096 May  7 14:11 tmp
-rw-r--r-- 1 cssmuadm cssmuadm 2473400 May  7 14:12 world192.txt
cssmuadm@lnx:~$
```

# Standard Input and Output: Redirection

Double greater than sign (>>) also redirects STDOUT to a file. It creates a new file if not present, otherwise appends to it.

While  singe greater than sign (>) always overwrites file

cssmuadm@lnx: ~

```
cssmuadm@lnx:~$ ls
di.c  tmp  world192.txt
cssmuadm@lnx:~$ echo "line" >> out.txt
cssmuadm@lnx:~$ cat out.txt
line
cssmuadm@lnx:~$ echo "line" >> out.txt
cssmuadm@lnx:~$ cat out.txt
line
line
cssmuadm@lnx:~$ echo "line" > out.txt
cssmuadm@lnx:~$ cat out.txt
line
cssmuadm@lnx:~$ echo "line1" 1> out.txt
cssmuadm@lnx:~$ cat out.txt
line1
cssmuadm@lnx:~$ echo "line2" 1>> out.txt
cssmuadm@lnx:~$ ls
di.c  out.txt  tmp  world192.txt
cssmuadm@lnx:~$ cat out.txt
line1
line2
cssmuadm@lnx:~$
```

# Standard Input and Output: Redirection

To redirect STDERR use **2>** (or **2>>** to append)

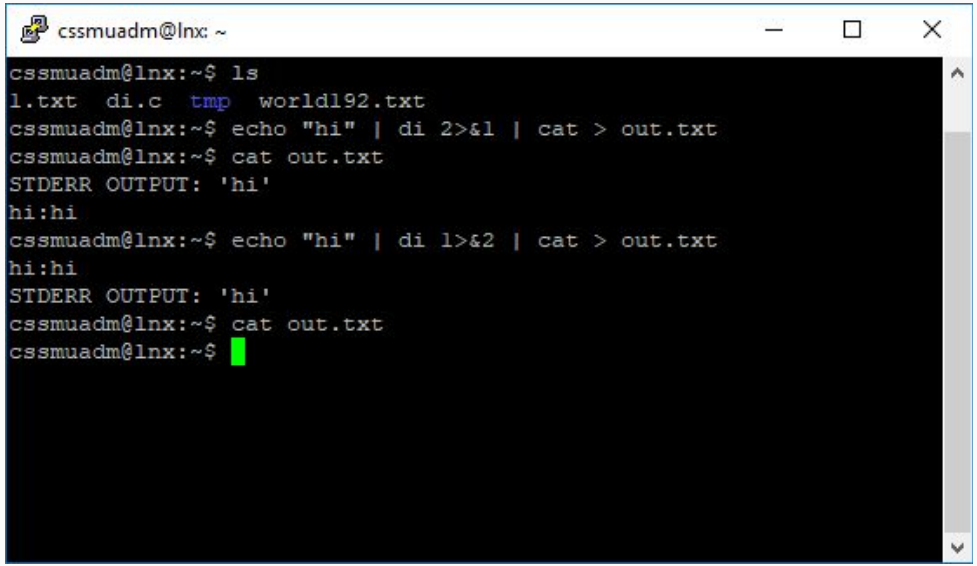
To redirect both STDOUT and STDERR use **&>**

```
cssmuadm@lnx: ~
cssmuadm@lnx:~$ echo "hi" | di
hi:hi
STDERR OUTPUT: 'hi'
cssmuadm@lnx:~$ echo "hi" | di 1> out.txt
STDERR OUTPUT: 'hi'
cssmuadm@lnx:~$ cat out.txt
hi:hi
cssmuadm@lnx:~$ echo "hi" | di 2> out.txt
hi:hi
cssmuadm@lnx:~$ cat out.txt
STDERR OUTPUT: 'hi'
cssmuadm@lnx:~$ echo "hi" | di &> out.txt
cssmuadm@lnx:~$ cat out.txt
STDERR OUTPUT: 'hi'
hi:hi
cssmuadm@lnx:~$
```

# Standard Input and Output: Redirection

To redirect STDERR to STDOUT use
**2>&1**

To redirect STDOUT to STDERR use
**1>&2**



# Standard Input and Output: Redirection

To redirect something to nowhere (useful in scripts), use redirection to **/dev/null**

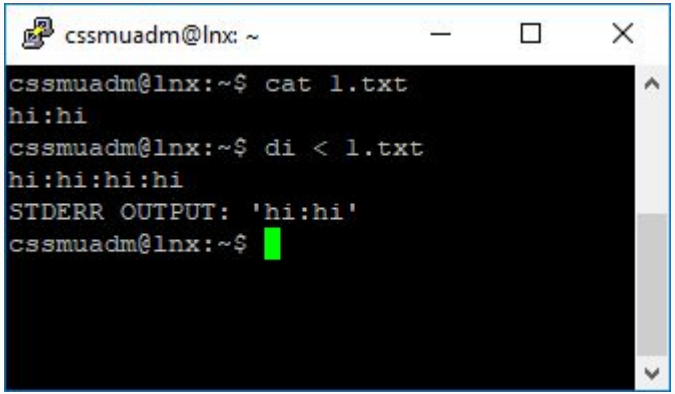
You can redirect STDERR to one file and STDOUT to another file

```
cssmuadm@lnx: ~
cssmuadm@lnx:~$ ls
di.c  out.txt  tmp  world192.txt
cssmuadm@lnx:~$ echo "hi" | di 2> /dev/null
hi:hi
cssmuadm@lnx:~$ echo "hi" | di 2> /dev/null 1> 1.txt
cssmuadm@lnx:~$ cat 1.txt
hi:hi
cssmuadm@lnx:~$
```


[link]

# Standard Input and Output: Redirection

Less than sign (<) is used to accept input from file



# Standard Input and Output: Summary

- Pipes (|) are used to connect output of one program with input of another program

- Redirection is used to redirect STDOUT/STDERR to file (1>, 2>) OR use contents of file as STDIN (<)

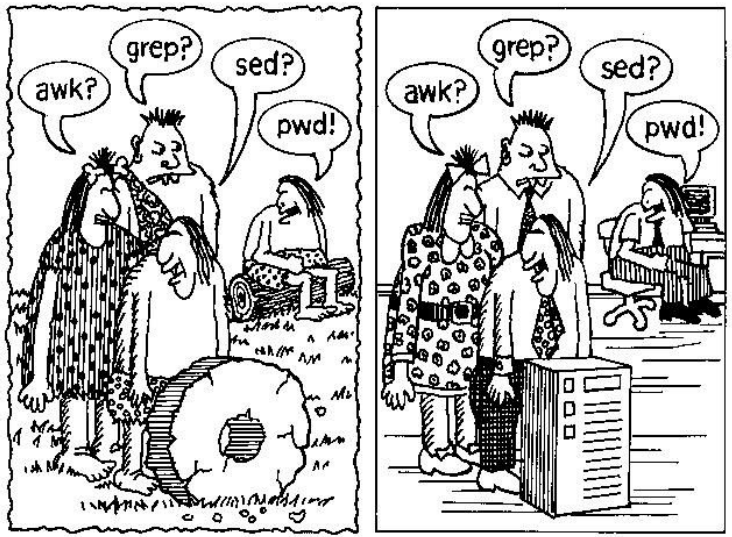
# Exercise

*List content of **/bin** directory to file with name **binlist.txt***

# Working with text, regular expressions

There is a set of programs that can get text data from STDIN (or as an argument) and transform this text data someway:

- sort
- grep
- awk
- tr
- sed
- shuf
- wc  ...etc




[link]

# sort

**sort** - sort lines of text files

- -n does numerical sort rather than string comparison
- -r does reverse sort
- -t specifies field separator (e.g., *sort -t','*)
- -k sort by key (usually field number goes after key like *k2*)

root@lnx: /home/student

```
root@lnx:/home/student# ls | sort | head
a_cornish
a_hassan
a_sharma
a_sivaraman
b_ahuja
bb_verma
b_shree
ca_irfanoglu
c_mccavour
d_dsouzabhatt
root@lnx:/home/student# ls | sort -r | head
z_yan
x_tong
v_govindan
v_boopathy
s_thapamagar
s_punnoli
s_padikar
s_magray
sk_channaveerabhadraiah
sh_sadi
root@lnx:/home/student#
```

# numeric sort vs lexicographical sort

cssmuadm@lnx: /bin

```
cssmuadm@lnx:/bin$ du -s * | sort -r | head
96      ps
96      gzip
96      df
84      dumpkeys
8       znew
8       zgrep
8       zdiff
8       lesspipe
8       gzexe
8       bzexe
cssmuadm@lnx:/bin$ du -s * | sort -r -n | head
1920    busybox
1016    bash
664     networkctl
648     systemctl
512     btrfs
488     journalctl
444     loginctl
440     udevadm
376     tar
368     ip
cssmuadm@lnx:/bin$
```

# Exercise

*Sort list of students (/home/students directory) by student last name*

# grep

**grep** - print lines matching a pattern

Can work both as FILTER (accept STDIN) or with command line arguments

-R means recursively scan the directory from the command line argument

# grep

root@lnx: /home/student

```
root@lnx:/home/student# ls | grep an
a_hassan
a_sivaraman
ca_irfanoglu
dk_sambhwani
j_laplante
j_wang
kk_murugappan
p_bandyopadhyay
sk_channaveerabhadraiah
v_govindan
z_yan
root@lnx:/home/student# ls | grep an$
a_hassan
a_sivaraman
kk_murugappan
v_govindan
z_yan
root@lnx:/home/student# grep -R ERROR /var/log
/var/log/apport.log.1:ERROR: apport (pid 6549) Sat Jan 12 17:42:42 2019: called for pid 6548, signal 11, core limit 0,
 dump mode 1
/var/log/apport.log.1:ERROR: apport (pid 6549) Sat Jan 12 17:42:42 2019: executable: /lib/klibc-k3La8MUnuzHQ0_kG8hokcG
AC0PA.so (command line "/lib/klibc-k3La8MUnuzHQ0_kG8hokcGAC0PA.so")
/var/log/apport.log.1:ERROR: apport (pid 6549) Sat Jan 12 17:42:42 2019: is_closing_session(): no DBUS_SESSION_BUS_ADD
RESS in environment
/var/log/apport.log.1:ERROR: apport (pid 6549) Sat Jan 12 17:42:42 2019: wrote report /var/crash/_lib_klibc-k3La8MUnuz
HQ0_kG8hokcGAC0PA.so.1028.crash
^C
root@lnx:/home/student#
```

# grep

**-v** flag INVERTS match (finds lines NOT matching pattern)

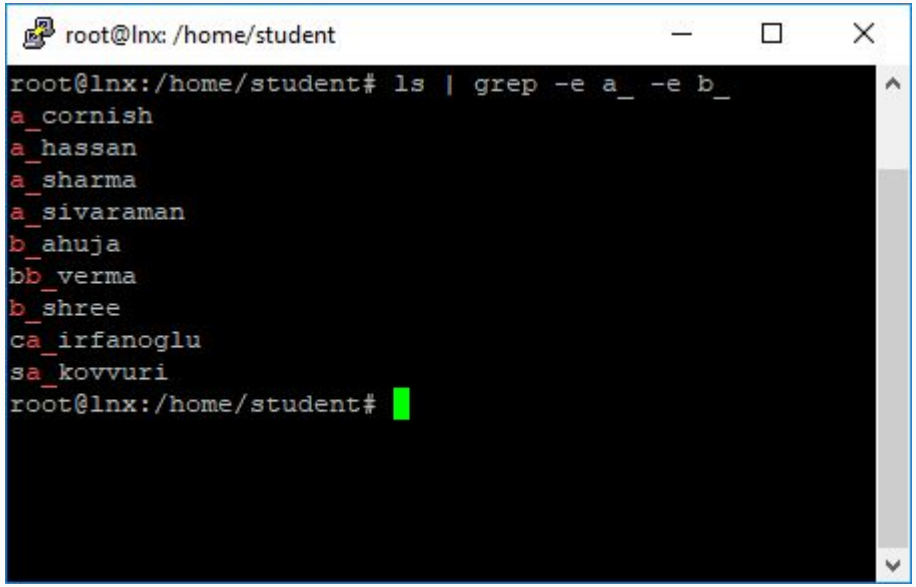
root@lnx: /home/student

```
root@lnx:/home/student# ls | grep -v s_
a_cornish
a_hassan
a_sharma
a_sivaraman
b_ahuja
bb_verma
b_shree
ca_irfanoglu
c_mccavour
d_dsouzabhatt
dk_sambhwani
d_malone
g_singh
j_kour
j_laplante
j_wang
j_weston
kk_murugappan
m_fasuyi
mk_baria
mm_leong
mn_hussain
p_bandyopadhyay
q_cai
r_gupta
r_nomula
sa_kovvuri
sh_sadi
sk_channaveerabhadraiah
v_boopathy
v_govindan
x_tong
z_yan
root@lnx:/home/student#
```

# grep

-e PATTERN -e PATTERN -e PATTERN

Match many patterns at a time



# grep as much as you need

root@lnx: /home/student

```
root@lnx:/home/student# ls | grep -e a_ -e b_
a_cornish
a_hassan
a_sharma
a_sivaraman
b_ahuja
bb_verma
b_shree
ca_irfanoglu
sa_kovvuri
root@lnx:/home/student# ls | grep -e a_ -e b_ | grep -v s
b_ahuja
bb_verma
ca_irfanoglu
root@lnx:/home/student#
```

# grep

-m flag LIMITS number of output lines per file

root@lnx: /home/student

```
root@lnx:/home/student# ls | grep b -m 4
b_ahuja
bb_verma
b_shree
d_dsouzabhatt
root@lnx:/home/student# ls | grep b -m 5
b_ahuja
bb_verma
b_shree
d_dsouzabhatt
dk_sambhwani
root@lnx:/home/student# ls | grep b -m 10
b_ahuja
bb_verma
b_shree
d_dsouzabhatt
dk_sambhwani
mk_baria
p_bandyopadhyay
rs_oberoi
sk_channaveerabhadraiah
v_boopathy
root@lnx:/home/student#
```

# grep

-A *N* print N lines AFTER match

-B *N* print N lines BEFORE match

use case: analyzing logs

```
root@lnx: /home/student
root@lnx:/home/student# grep -A 2 -B 2 j_salvi /var/log/auth*
/var/log/auth.log-May  8 22:16:29 lnx systemd: pam_unix(systemd-user:session): session opened for user v_nagamanickam by (uid=0
)
/var/log/auth.log-May  8 22:16:29 lnx su[29687]: pam_systemd(su:session): Cannot create session: Already running in a session
/var/log/auth.log:May  8 22:16:29 lnx sshd[29572]: Accepted password for j_salvi from 140.184.16.150 port 50464 ssh2
/var/log/auth.log:May  8 22:16:29 lnx sshd[29572]: pam_unix(sshd:session): session opened for user j_salvi by (uid=0)
/var/log/auth.log:May  8 22:16:29 lnx systemd: pam_unix(systemd-user:session): session opened for user j_salvi by (uid=0)
/var/log/auth.log:May  8 22:16:29 lnx systemd-logind[757]: New session 560 of user j_salvi.
/var/log/auth.log-May  8 22:16:41 lnx sshd[29802]: Invalid user Dauneo from 140.184.17.209
/var/log/auth.log-May  8 22:16:41 lnx sshd[29802]: input_userauth_request: invalid user Dauneo [preauth]
--
/var/log/auth.log-May  9 01:45:39 lnx systemd-logind[757]: Removed session 561.
/var/log/auth.log-May  9 01:45:39 lnx systemd: pam_unix(systemd-user:session): session closed for user r_mishra
/var/log/auth.log:May  9 01:48:23 lnx sshd[29572]: pam_unix(sshd:session): session closed for user j_salvi
/var/log/auth.log-May  9 01:48:23 lnx systemd-logind[757]: Removed session 560.
/var/log/auth.log:May  9 01:48:23 lnx systemd: pam_unix(systemd-user:session): session closed for user j_salvi
/var/log/auth.log-May  9 01:49:29 lnx sshd[29251]: pam_unix(sshd:session): session closed for user o_selere
/var/log/auth.log-May  9 01:49:29 lnx systemd-logind[757]: Removed session 551.
--
/var/log/auth.log-May 10 22:40:43 lnx sshd[21184]: pam_unix(sshd:session): session opened for user m_maruf by (uid=0)
/var/log/auth.log-May 10 22:40:43 lnx systemd-logind[757]: New session 779 of user m_maruf.
/var/log/auth.log:May 10 22:40:44 lnx sshd[21182]: Accepted password for j_salvi from 140.184.17.227 port 50874 ssh2
/var/log/auth.log:May 10 22:40:44 lnx sshd[21182]: pam_unix(sshd:session): session opened for user j_salvi by (uid=0)
/var/log/auth.log:May 10 22:40:44 lnx systemd-logind[757]: New session 780 of user j_salvi.
/var/log/auth.log:May 10 22:40:44 lnx systemd: pam_unix(systemd-user:session): session opened for user j_salvi by (uid=0)
/var/log/auth.log:May 10 22:40:54 lnx sshd[21258]: Accepted password for j_salvi from 140.184.17.227 port 50875 ssh2
/var/log/auth.log:May 10 22:40:54 lnx sshd[21258]: pam_unix(sshd:session): session opened for user j_salvi by (uid=0)
/var/log/auth.log:May 10 22:40:54 lnx systemd-logind[757]: New session 781 of user j_salvi.
/var/log/auth.log-May 10 22:41:27 lnx sshd[21386]: Accepted password for y_yin from 140.184.206.20 port 55180 ssh2
/var/log/auth.log-May 10 22:41:27 lnx sshd[21386]: pam_unix(sshd:session): session opened for user y_yin by (uid=0)
--
/var/log/auth.log-May 11 00:52:10 lnx systemd-logind[757]: Removed session 776.
/var/log/auth.log-May 11 00:55:03 lnx sshd[24046]: Connection closed by 104.131.202.23 port 42504 [preauth]
/var/log/auth.log:May 11 00:56:16 lnx sshd[21182]: pam_unix(sshd:session): session closed for user j_salvi
/var/log/auth.log-May 11 00:56:16 lnx systemd-logind[757]: Removed session 780.
/var/log/auth.log-May 11 00:58:01 lnx sshd[24050]: Did not receive identification string from 67.42.27.101
--
/var/log/auth.log-May 11 01:39:01 lnx CRON[24177]: pam_unix(cron:session): session opened for user root by (uid=0)
/var/log/auth.log-May 11 01:39:01 lnx CRON[24177]: pam_unix(cron:session): session closed for user root
/var/log/auth.log:May 11 01:42:08 lnx sshd[21258]: pam_unix(sshd:session): session closed for user j_salvi
/var/log/auth.log-May 11 01:42:08 lnx systemd-logind[757]: Removed session 781.
```

# case-insensitive grep

-i is case-insensitive

root@lnx: /var/data

```
root@lnx:/var/data# tail alice29.txt
she would keep, through all her riper years, the simple and
loving heart of her childhood:  and how she would gather about
her other little children, and make THEIR eyes bright and eager
with many a strange tale, perhaps even with the dream of
Wonderland of long ago:  and how she would feel with all their
simple sorrows, and find a pleasure in all their simple joys,
remembering her own child-life, and the happy summer days.

                              THE END
root@lnx:/var/data# tail alice29.txt | grep -i their
her other little children, and make THEIR eyes bright and eager
Wonderland of long ago:  and how she would feel with all their
simple sorrows, and find a pleasure in all their simple joys,
root@lnx:/var/data#
```

# Exercise

*Using **grep**, find from which ip address you accessed the lnx server last time*

*logfiles: **/var/log/auth*** (you have temporary read permissions)*

TIP: **sshd** keyword can help you to filter records that contain IP address info. Your **username** is also an obvious filter

# Regex basics

It's something like **"NORMAL TEXT + some special symbols"** that MATCHES text

STRING: BANANA

Regex 1: BANANA

Regex 1: ^BANANA$

Regex 2: B.NA.A

Regex 3: BAN...

Regex 4: BO.*A //does not match

Regex 5: B.*A

Regex 6: .*

# ^Regex$

^ matches BEGINNING OF LINE

$ matches END OF LINE

root@lnx: /home/student

```
root@lnx:/home/student# ls | grep a_
a_cornish
a_hassan
a_sharma
a_sivaraman
ca_irfanoglu
sa_kovvuri
root@lnx:/home/student# ls | grep ^a_
a_cornish
a_hassan
a_sharma
a_sivaraman
root@lnx:/home/student# ls | grep an
a_hassan
a_sivaraman
ca_irfanoglu
dk_sambhwani
j_laplante
j_wang
kk_murugappan
p_bandyopadhyay
sk_channaveerabhadraiah
v_govindan
z_yan
root@lnx:/home/student# ls | grep an$
a_hassan
a_sivaraman
kk_murugappan
v_govindan
z_yan
root@lnx:/home/student#
```

# \bRegex\b

\b is a word boundary

root@lnx: /var/data

```
root@lnx:/var/data# tail alice29.txt
she would keep, through all her riper years, the simple and
loving heart of her childhood:  and how she would gather about
her other little children, and make THEIR eyes bright and eager
with many a strange tale, perhaps even with the dream of
Wonderland of long ago:  and how she would feel with all their
simple sorrows, and find a pleasure in all their simple joys,
remembering her own child-life, and the happy summer days.

                              THE END
root@lnx:/var/data# tail alice29.txt | grep \\bwould\\b
she would keep, through all her riper years, the simple and
loving heart of her childhood:  and how she would gather about
Wonderland of long ago:  and how she would feel with all their
root@lnx:/var/data# tail alice29.txt | grep \\bwo..d\\b
she would keep, through all her riper years, the simple and
loving heart of her childhood:  and how she would gather about
Wonderland of long ago:  and how she would feel with all their
root@lnx:/var/data# tail alice29.txt | grep \\bwo\\b
root@lnx:/var/data# tail alice29.txt | grep wo
she would keep, through all her riper years, the simple and
loving heart of her childhood:  and how she would gather about
Wonderland of long ago:  and how she would feel with all their
root@lnx:/var/data#
```

# R..ex

. (DOT) matches ANY **single** character

You can use **N** dots to match any **N** characters

NOTE: -E flag stays for Extended, can always be used

root@lnx: /var/data

```
root@lnx:/var/data# tail alice29.txt
she would keep, through all her riper years, the simple and
loving heart of her childhood:  and how she would gather about
her other little children, and make THEIR eyes bright and eager
with many a strange tale, perhaps even with the dream of
Wonderland of long ago:  and how she would feel with all their
simple sorrows, and find a pleasure in all their simple joys,
remembering her own child-life, and the happy summer days.

                              THE END
root@lnx:/var/data# tail alice29.txt | grep pl...ure
simple sorrows, and find a pleasure in all their simple joys,
root@lnx:/var/data# tail alice29.txt | grep pl.a.ure
simple sorrows, and find a pleasure in all their simple joys,
root@lnx:/var/data# tail alice29.txt | grep pl.{4}re
root@lnx:/var/data# tail alice29.txt | grep -E pl.{4}re
simple sorrows, and find a pleasure in all their simple joys,
root@lnx:/var/data#
```

# R+ege{1}xX*

* (STAR) works as a quantifier: matches ANY number of character **specified right before it**. Examples:

- .* means any number of any character (matches everything)
- a* means any number of **a** (including zero)

+ (PLUS) also works as a quantifier: matches **one or more number** of character specified right before it

- {N} to match **exactly N** characters
- {N,} to match **N and more** characters
- {,N} to match **up to N** characters
- {M,N} to match **from M to N** characters

root@lnx: /home/student

```
root@lnx:/home/student# ls
a_cornish       dk_sambhwani    mm_leong          sk_channaveerabhadraiah
a_hassan        d_malone        mn_hussain        s_magray
a_sharma        g_singh         p_bandyopadhyay   s_padikar
a_sivaraman     j_kour          q_cai             s_punnoli
b_ahuja         j_laplante      r_gupta           s_thapamagar
bb_verma        j_wang          r_nomula          v_boopathy
b_shree         j_weston        rs_oberoi         v_govindan
ca_irfanoglu    kk_murugappan   sa_kovvuri        x_tong
c_mccavour      m_fasuyi        s_alachkar        z_yan
d_dsouzabhatt   mk_baria        sh_sadi
root@lnx:/home/student# ls | grep ^s.*i$
sa_kovvuri
sh_sadi
s_punnoli
root@lnx:/home/student# ls | grep -E ^sk_chan*aveerabhadraiah$
sk_channaveerabhadraiah
root@lnx:/home/student# ls | grep -E ^sk_chan+aveerabhadraiah$
sk_channaveerabhadraiah
root@lnx:/home/student# ls | grep -E ^sk_channZ*aveerabhadraiah$
sk_channaveerabhadraiah
root@lnx:/home/student# ls | grep -E ^sk_channZ+aveerabhadraiah$
root@lnx:/home/student# ls | grep -E ^sk_chan{2}aveerabhadraiah$
sk_channaveerabhadraiah
root@lnx:/home/student#
```

# R[a-z]gex

[abc] set of symbols in SQUARE BRACKETS means that **any** of the symbols can be matched

Special cases:

[0-9] - any digit

[a-z] - any lowercase letter

[A-Z] -any uppercase letter

root@lnx: /home/student

```
root@lnx:/home/student# ls
a_cornish       dk_sambhwani   mm_leong          sk_channaveerabhadraiah
a_hassan        d_malone       mn_hussain        s_magray
a_sharma        g_singh        p_bandyopadhyay   s_padikar
a_sivaraman     j_kour         q_cai             s_punnoli
b_ahuja         j_laplante     r_gupta           s_thapamagar
bb_verma        j_wang         r_nomula          v_boopathy
b_shree         j_weston       rs_oberoi         v_govindan
ca_irfanoglu    kk_murugappan  sa_kovvuri        x_tong
c_mccavour      m_fasuyi       s_alachkar        z_yan
d_dsouzabhatt   mk_baria       sh_sadi
root@lnx:/home/student# ls | grep -E ^[mq]+_.*$
m_fasuyi
mm_leong
q_cai
root@lnx:/home/student# ls | grep -E ^[mq]_.*$
m_fasuyi
q_cai
root@lnx:/home/student# ls | grep -E ^[xz]_.*$
x_tong
z_yan
root@lnx:/home/student#
```

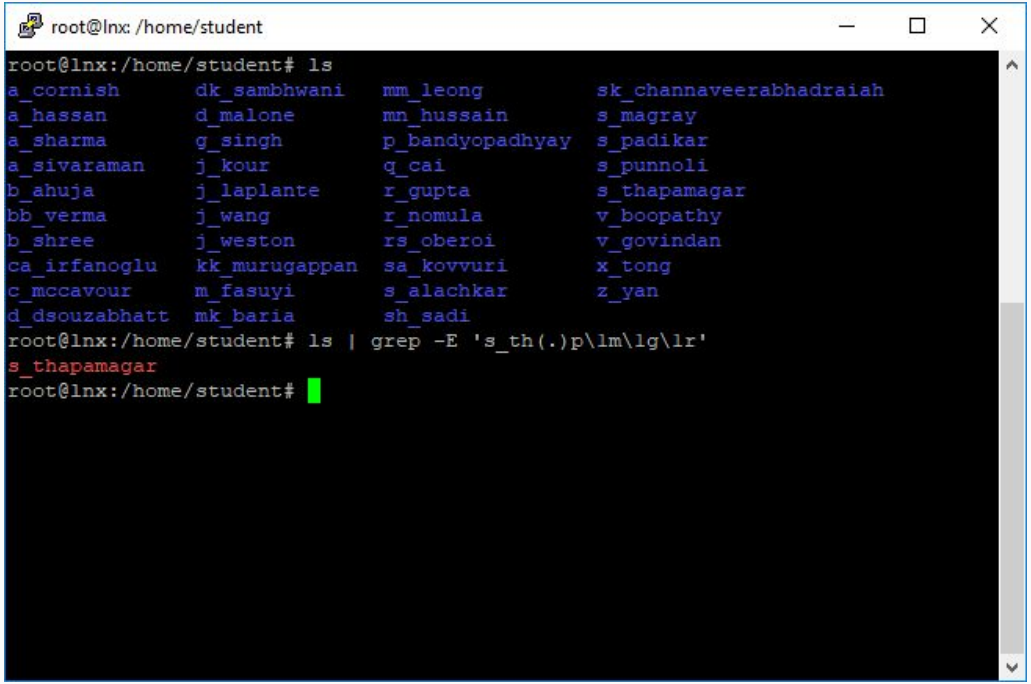
# R(e)g\1x

() - everything between parentheses is a **capturing group**

*Q: Why you need them?*

*A: To reference what's between them again using a **backreference***

Backreference is a backslash followed by capturing group number (e.g., \1, \2, \3 … \N) where N is number of capturing groups in your regex



# Regex\\.

What if we need to match a "**special**" character that have meaning in regex (e.g., ".", "*", "$", "?", "^")?

Just escape them with \\

```
cssmuadm@lnx: ~
cssmuadm@lnx:~$ cat test.doc
hello hello . how ORE you..?
How much $ do you have?
hehe ^_^
cssmuadm@lnx:~$ cat test.doc | grep -E .
hello hello . how ORE you..?
How much $ do you have?
hehe ^_^
cssmuadm@lnx:~$ cat test.doc | grep -E \\.
hello hello . how ORE you..?
cssmuadm@lnx:~$ cat test.doc | grep -E $
hello hello . how ORE you..?
How much $ do you have?
hehe ^_^
cssmuadm@lnx:~$ cat test.doc | grep -E \\$
How much $ do you have?
cssmuadm@lnx:~$ cat test.doc | grep -E \\^
hehe ^_^
cssmuadm@lnx:~$
```

# Exercise

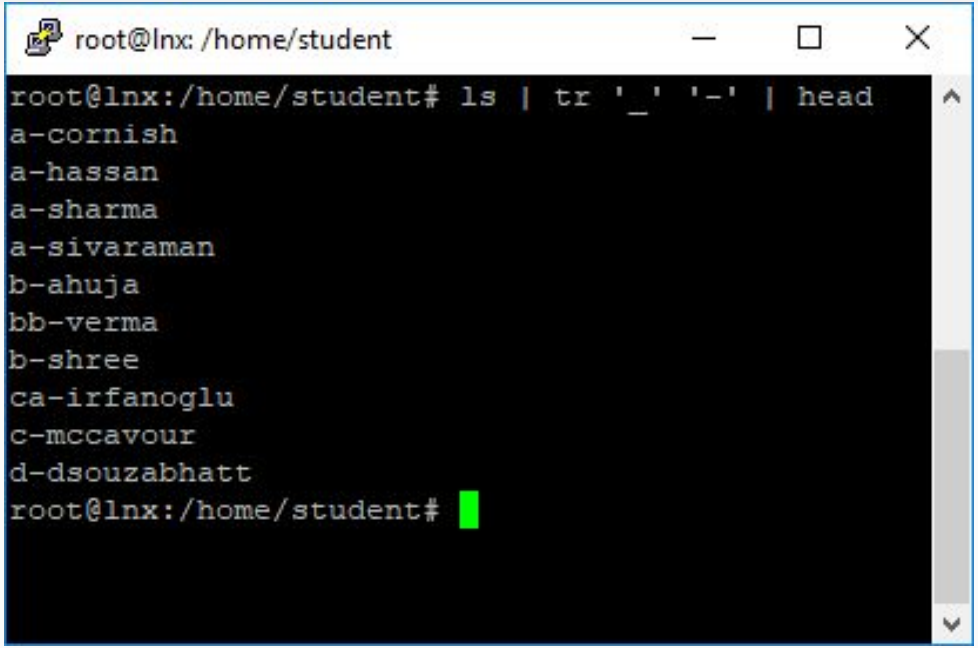
*Go to /home/student*

- *Use **ls + grep** output your and your neighbour's account together*
- *Using **ls + grep** find all students with double letters in their last name*

# tr

tr - translate or delete characters

tr [character to replace] [replacement]



# tr

root@lnx: /home/student

```
root@lnx:/home/student# ls | tr '_' '-' | head
a-cornish
a-hassan
a-sharma
a-sivaraman
b-ahuja
bb-verma
b-shree
ca-irfanoglu
c-mccavour
d-dsouzabhatt
root@lnx:/home/student# ls | tr '_' '-' | head | tr '[abc]' '_'
_-_ornish
_-h_ss_n
_-sh_rm_
_-siv_r_m_n
_-_huj_
__-verm_
_-shree
__-irf_noglu
_-m___vour
d-dsouz__h_tt
root@lnx:/home/student#
```

# tr

tr -d [abcd]

-d means DELE

root@lnx: /home/student

```
root@lnx:/home/student# ls | tr -d '_' | tail
shsadi
skchannaveerabhadraiah
smagray
spadikar
spunnoli
sthapamagar
vboopathy
vgovindan
xtong
zyan
root@lnx:/home/student#
```

# Exercise

*Go to /home/student*

*Use **ls + grep** to get your account and **tr** to transform your account name to UPPERCASE*

Hint: [:lower:] is all lowercase and [:upper:] is all uppercase

# ‘s/sad/sed/g’

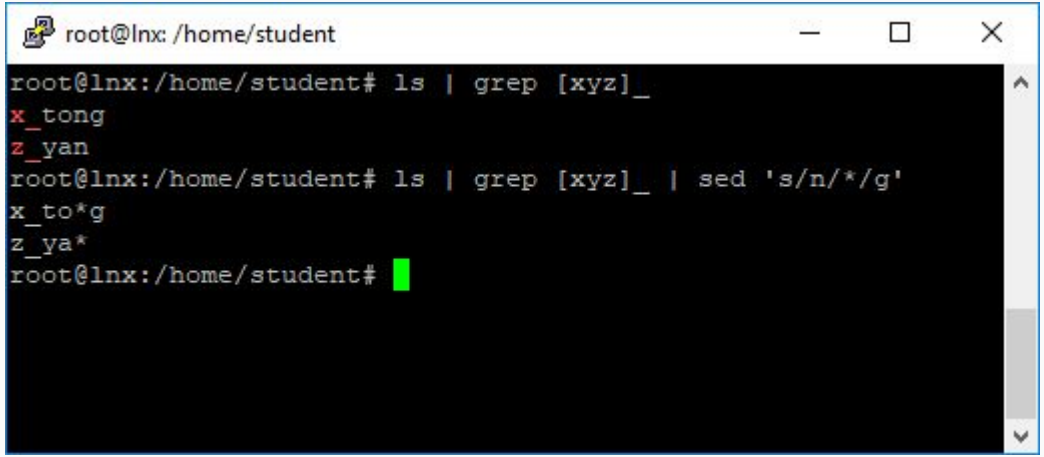
**sed** - stream editor for filtering and transforming

Common structure:

**sed -r 's/to_replace/replacement/g'**

-r tells to use extended regular expressions

g at the end tells to replace **all** occurrences rather than just first one



# ‘s/sa(.)/se\1/g’

root@lnx: /home/student

```
root@lnx:/home/student# ls | tail
sh_sadi
sk_channaveerabhadraiah
s_magray
s_padikar
s_punnoli
s_thapamagar
v_boopathy
v_govindan
x_tong
z_yan
root@lnx:/home/student# ls | tail | sed -r 's/^(.*)_(.*)$/\2\_\1/g'
sadi_sh
channaveerabhadraiah_sk
magray_s
padikar_s
punnoli_s
thapamagar_s
boopathy_v
govindan_v
tong_x
yan_z
root@lnx:/home/student#
```

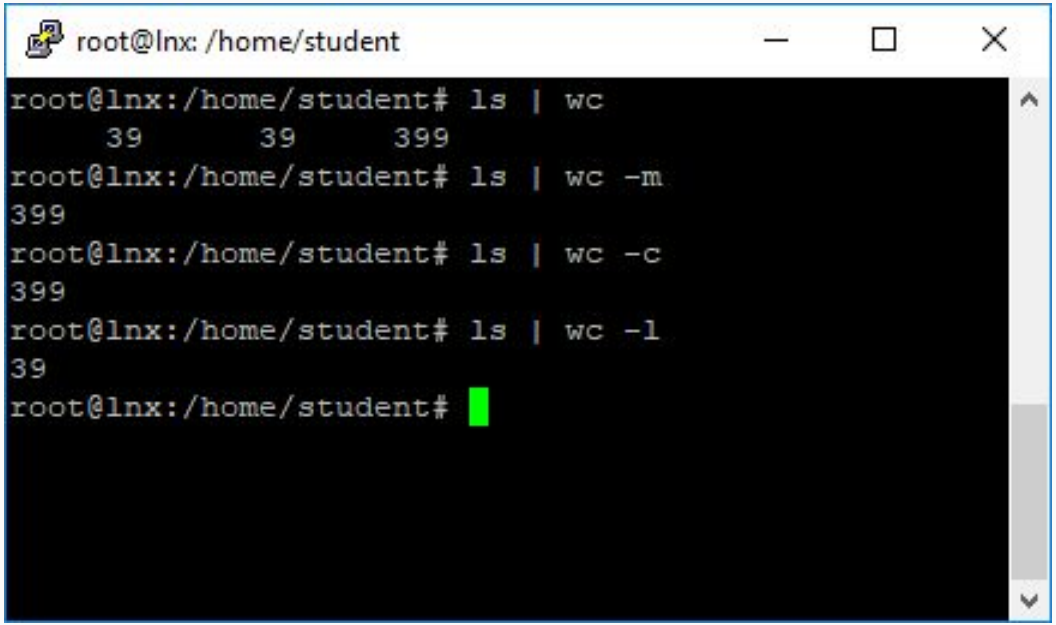
# wc != restroom

**wc** - print newline, word, and byte counts for each file

-c --bytes print the byte counts

-m --chars print the character counts

-l --lines print the newline counts



# shuf

**shuf** - generate random permutations (shuffle lines)

root@lnx: /home/student

```
root@lnx:/home/student# ls | shuf | head
sa_kovvuri
j_wang
c_mccavour
d_dsouzabhatt
p_bandyopadhyay
ca_irfanoglu
z_yan
d_malone
s_punnoli
b_ahuja
root@lnx:/home/student# ls | shuf | head
sa_kovvuri
ca_irfanoglu
b_shree
x_tong
s_punnoli
v_boopathy
c_mccavour
q_cai
g_singh
sh_sadi
root@lnx:/home/student#
```

# split

**split** - split a file into pieces

-l flag: number of lines in splitted files

-n flag: number of chunks. Useful with option -n l/N where *N* number of files: will split into *N* files without splitting lines/records

root@lnx: ~/split

```
root@lnx:~# mkdir split
root@lnx:~# cd split/
root@lnx:~/split# ls /home/student | split -l 10
root@lnx:~/split# ls -l
total 16
-rw-r--r-- 1 root root 103 Jan 17 17:34 xaa
-rw-r--r-- 1 root root  96 Jan 17 17:34 xab
-rw-r--r-- 1 root root  99 Jan 17 17:34 xac
-rw-r--r-- 1 root root 101 Jan 17 17:34 xad
root@lnx:~/split# cat xaa
a_cornish
a_hassan
a_sharma
a_sivaraman
b_ahuja
bb_verma
b_shree
ca_irfanoglu
c_mccavour
d_dsouzabhatt
root@lnx:~/split# wc -l *
 10 xaa
 10 xab
 10 xac
  9 xad
 39 total
root@lnx:~/split#
```

# awk

"**AWK** is a language for processing text files. A file is treated as a sequence of records, and by default each line is a record. Each line is broken up into a sequence of fields, so we can think of the first word in a line as the first field, the second word as the second field, and so on. **An AWK program is a sequence of pattern-action statements**. AWK reads the input a line at a time. A line is scanned for each pattern in the program, and for each pattern that matches, the associated action is executed." - Alfred V. Aho[1]

Most used feature:

**awk '{print $1, $2}'**

where $1 $2 are numbers of columns/fields

Flag -F',' is used to specify field separator

root@lnx: /home/student

```
root@lnx:/home/student# ls -l | tail -5
drwxr-x--x 3 s_thapamagar          s_thapamagar          4096 Jan 14 21:20 s_thapamagar
drwxr-x--x 2 v_boopathy            v_boopathy            4096 Jan 14 13:18 v_boopathy
drwxr-x--x 2 v_govindan            v_govindan            4096 Jan 14 13:18 v_govindan
drwxr-x--x 2 x_tong                x_tong                4096 Jan 17 14:11 x_tong
drwxr-x--x 2 z_yan                 z_yan                 4096 Jan 14 13:18 z_yan
root@lnx:/home/student# ls -l | tail -5 | awk '{print $3}'
s_thapamagar
v_boopathy
v_govindan
x_tong
z_yan
root@lnx:/home/student# ls -l | tail -5 | awk '{print $4}'
s_thapamagar
v_boopathy
v_govindan
x_tong
z_yan
root@lnx:/home/student# ls -l | tail -5 | awk '{print $5}'
4096
4096
4096
4096
4096
root@lnx:/home/student#
```

# exit