MENGELOLA TEKS PADA LINUX

1

OBJEKTIF :

1. Mahasiswa Mampu Mengetahui Bagaimana Cara Mengelola Teks Pada Linux.
2. Mahasiswa Mampu Untuk Memahami Perintah Dasar Untuk Mengelola Teks Pada Linux.

---

1. PENDAHULUAN

Sebagian besar file dalam file system bertipe file teks. File teks hanya berisi teks, tidak ada fitur pemformatan seperti yang anda lihat dalam file pengolah kata.

Karena ada banyak file pada sistem Linux, sejumlah besar perintah tersedia untuk membantu pengguna memanipulasi file teks. Ada perintah untuk melihat dan memodifikasi file-file ini dalam berbagai cara.

Selain itu, ada fitur yang tersedia dalam shell untuk mengontrol output perintah, jadi kita dapat langsung di alihkan ke file lain atau perintah lain dibandingkan di tempatkan di jendela terminal. Fitur pengalihan ini memberi pengguna lingkungan yang jauh lebih fleksibel dan kuat untuk bekerja di dalamnya.

1.1 MELIHAT FILE DI TERMINAL

Perintah `cat` merupakan singkatan dari concatenate, yang mana perintah ini adalah perintah yang berfungsi untuk membuat dan menampilkan file teks, serta menggabungkan salinan file teks. Salah satu kegunaan cat yang paling populer adalah untuk menampilkan konten file teks. Untuk menampilkan

file dalam output standar, gunakan perintah cat, ketik perintah cat lalu diikuti dengan nama file:

```
sysadmin@localhost:~$ cd Documents
sysadmin@localhost:~/Documents$ cat food.txt
Food is good.
```

Meskipun terminal adalah output default dari perintah ini, perintah `cat` juga dapat digunakan untuk mengarahkan konten file ke file lain atau input untuk perintah lain dengan menggunakan karakter *redirection*.

1.2 MELIHAT FILE MENGGUNAKAN PAGER

Perintah `cat` ideal untuk file berukuran kecil, namun tidak ideal untuk file berukuran besar. Perintah `cat` tidak menyediakan cara yang mudah untuk menjeda dan memulai ulang tampilan, sehingga seluruh konten file dibuang ke layar.

Untuk file yang lebih besar, gunakan perintah `pager`. Perintah `pager` menampilkan satu halaman data pada satu waktu, memungkinkan kita untuk bergerak maju dan mundur dalam file dengan menggunakan tombol gerakan.

Ada dua perintah pager yang umum digunakan:

- Perintah `less` memberikan kemampuan paging yang sangat baik. Biasanya pager default yang digunakan oleh perintah. Seperti perintah man.
- Perintah `more` telah ada pada UNIX. Meskipun ia memiliki fitur tidak sebaik perintah yang less, namun, perintah less tidak disertakan dalam semua distribusi Linux. Sedangkan perintah more ada dalam semua distribusi Linux.

Perintah more dan less menerima pengguna untuk pindah dokumen dengan menggunakan perintah keystroke. Karena developers menjadikan perintah `more` dan `less` sebagai dasar fungsionalitas. Semua perintah keystroke yang tersedia di perintah more juga bekerja dalam perintah less.

Namun kita akan berfokus pada perintah less. Perintah more masih dapat digunakan saat perintah less tidak tersedia. Perlu diingat bahwa perintah keystroke berfungsi di perintah less dan more.

1.2.1 PERINTAH PAGER MOVEMENT

Untuk melihat file dengan perintah `less` ketikkan nama file tersebut sebagai argument, seperti dibawah ini :

```
sysadmin@localhost:~/Documents$ less words
```

Ada banyak pergerakan perintah untuk `less`, masing-masing dengan beberapa peluang kunci atau kombinasi kunci. Meskipun ini mungkin tampak sulit, kita tidak perlu menghafal semua perintah. Saat melihat file dengan perintah `less`, gunakan tombol H atau Shift + H untuk menampilkan layar bantuan:

```
                  SUMMARY OF LESS COMMANDS


      Commands marked with * may be preceded by a number, N.
      Notes in parentheses indicate the behavior if N is given.
      A key preceded by a caret indicates the Ctrl key; thus ^K
is ctrl-K.


  h  H                 Display this help.
  q  :q  Q  :Q  ZZ     Exit.
 ---------------------------------------------------------------
---------

                           MOVING


  e  ^E  j  ^N  CR  *  Forward  one line   (or N lines).
  y  ^Y  k  ^K  ^P  *  Backward one line   (or N lines).
  f  ^F  ^V  SPACE  *  Forward  one window (or N lines).
  b  ^B  ESC-v      *  Backward one window (or N lines).
  z                 *  Forward  one window (and set window to N)
.
```

```
  w                 *  Backward one window (and set window to N)
.
  ESC-SPACE         *  Forward  one window, but don't stop at en
d-of-file.
  d  ^D             *  Forward  one half-window (and set half-wi
ndow to N).
  u  ^U             *  Backward one half-window (and set half-wi
ndow to N).
  ESC-)  RightArrow *  Left  one half screen width (or N positio
ns).
  ESC-(  LeftArrow  *  Right one half screen width (or N positio
ns).
HELP -- Press RETURN for more, or q when done
```

Kelompok pertama dari perintah gerakan yang akan difokuskan adalah yang paling umum digunakan. Untuk membuatnya lebih mudah, kunci identik `more` dan `less` dirangkum di bawah ini untuk menampilkan cara bergerak `more` dan `less` pada saat yang bersamaan:

Key	Movement
Spacebar	Window forward
B	Window backward
Enter	Line forward
Q	Exit
H	Help

Ketika menggunakan `less` sebagai *pager*, cara termudah untuk meneruskan halaman secara cepat adalah dengan menekan tombol ***Spacebar.***

1.2.2 PENCARIAN DATA MENGGUNAKAN PAGER

Ada 2 cara pencarian dalam perintah `less`: memajukan atau memundurkan posisi saat ini, untuk melakukannya, gunakan tombol slash (/), lalu ketikkan kalimat yang ingin dicari lalu tekan **Enter.**

```
Abdul
Abdul's
Abe
/frog
```

Jika kata ditemukan, maka kursor akan bergerak di dalam dokumen untuk mencocokkan. Sebagai contoh, kata frog ditemukan di dalam file words:

```
bullfrog
bullfrog's
bullfrogs
bullheaded
bullhorn
bullhorn's
```

Perhatikan bahwa “frog” bukan kata yang berdiri sendiri. Juga perhatikan Ketika perintah `less` berpindah ke kecocokkan pertama dari posisi saat ini, semua kecocokkan akan di *highlight*.

Jika tidak ada kecocokan yang dteruskan dari posisi saat ini, maka baris terakhir menampilkan kalimat `Pattern not found:`

```
Pattern not found (press RETURN)
```

Untuk menelusuri mundur dari posisi Anda saat ini, tekan tanda tanya (?) , lalu ketik teks atau pola yang akan dicocokkan dan tekan tombol Enter. Kursor bergerak mundur ke kecocokan pertama yang dapat ditemukannya atau melaporkan bahwa pola tidak dapat ditemukan.

Untuk melakukan *rollback*, tekan *question mark* (?) , lalu ketik teks atau pola yang ingin dicocokkan dan tekan tombol **Enter**. Kursor akan bergerak secara mundur ke kecocokan pertama atau melaporkan bahwa pola tidak dapat ditemukan.

Jika dalam pencarian di temukan lebih dari 1 kecocokkan, gunakan tombol N untuk memindahkan ke kecocokan berikutnya dan tekan tombol **Shift + N** untuk beralih ke kecocokan sebelumnya.

Istilah penelusuran sebenarnya menggunakan pola yang disebut Regular Expression (RegEx). Rincian lebih lanjut tentang ekspresi reguler disediakan nanti di bab ini.

1.3 HEAD DAN TAIL

Perintah `head` dan `tail` digunakan hanya untuk menampilkan beberapa baris pertama atau beberapa baris terakhir dari file, masing-masing (atau, bila digunakan dengan *pipe*, output dari perintah sebelumnya). Secara otomatis, perintah `head` dan `tail` menampilkan sepuluh baris file yang disediakan sebagai argumen.

Sebagai contoh, berikut ini tampilan perintah pada baris pertama pada file `/etc/sysctl.conf` :

```
sysadmin@localhost:~/Documents$ cd
sysadmin@localhost:~$ head /etc/sysctl.conf
#
# /etc/sysctl.conf - Configuration file for setting system variabl
es
# See /etc/sysctl.d/ for additional system variables
# See sysctl.conf (5) for information.
#

#kernel.domainname = example.com
```

```
# Uncomment the following to stop low-level messages on console
#kernel.printk = 3 4 1 3
```

Mengetikkan angka pada program akan menyebabkan perintah `head` dan `tail` menghasilkan jumlah baris yang ditentukan, yang mana baris tersebut tidak sesuai standardnya, yaitu 10. Misalnya kita akan menampilkan lima baris terakhir dari file /etc/sysctl.conf, maka ketikkan -5 pada program:

```
sysadmin@localhost:~$ tail -5 /etc/sysctl.conf
# Protects against creating or following links under certain condi
tions
# Debian kernels have both set to 1 (restricted)
# See https://www.kernel.org/doc/Documentation/sysctl/fs.txt
#fs.protected_hardlinks=0
#fs.protected_symlinks=0
```

`-n` juga dapat digunakan untuk mengindikasi berapa banyak baris dalam *output.* Ketikkan 3 sebagai argument :

```
sysadmin@localhost:~$ head -n 3 /etc/sysctl.conf
#
# /etc/sysctl.conf - Configuration file for setting system variabl
es
# See /etc/sysctl.d/ for additional system variables
```

Opsi Nilai Negatif

Secara tradisional di UNIX, jumlah baris dalam output ditentukan oleh jumlah baris yang kita masukkan sebagai opsi dalam program, jadi -3 berarti menunjukan 3 baris. Untuk perintah `tail`, baik `-3` atau `-n -3` tetap akan menunjukkan 3 baris pada *output* program.

Namun, perintah `head` dalam GNU mengenali `-n -3` sebagai perintah untuk menampilkan seluruh baris kecuali 3 baris terakhir dan perintah `head` juga mengenali `-3` jika ingin menampilkan 3 baris pertama.

Opsi Nilai Positif

Dalam GNU, perintah `tail` menerima variasi untuk menspesifikasikan angka pada baris yang akan di cetak. Jika `-n` digunakan dengan angka yang diawali dengan tanda tambah, maka perintah `tail` mengenalinya untuk menampilkan data mulai dari baris yang ditentukan dan terus berlanjut sampai akhir.

Sebagai contoh, tampilan dibawah ini memuat data dari `/etc/passwd` mulai dari baris 25 hingga akhir pada sebuah file :

`/etc/passwd` adalah file yang menyimpan informasi lain tentang id dan shell pengguna yang dapat dibaca oleh semua pengguna agar sistem berfungsi.

```
sysadmin@localhost:~$ nl /etc/passwd | tail -n +25
    25  sshd:x:103:65534::/var/run/sshd:/usr/sbin/nologin
    26  operator:x:1000:37::/root:/bin/sh
    27  sysadmin:x:1001:1001:System Administrator,,,,:/home/sysadm
in:/bin/bash
```

Pertimbangkan Hal Ini :

Perubahan file langsung dapat dilihat dengan menggunakan opsi `-f` pada perintah `tail` — berguna ketika Anda ingin melihat perubahan pada file secara *realtime*.

Contoh bagusnya adalah saat melihat file log sebagai administrator sistem. File log dapat digunakan untuk memecahkan masalah dan administrator sering

melihatnya “secara interaktif" dengan perintah `tail` saat menjalankan perintah di jendela terpisah.

Sebagai contoh, jika Anda masuk sebagai *user* root, Anda dapat memecahkan masalah dengan server email dengan melihat perubahan langsung ke file log `/var/log/mail.log`.

2. COMMAND LINE PIPES

Karakter *pipe* (|) dapat digunakan untuk mengirim *output* dalam suatu perintah ke perintah lain. Umumnya, Ketika *output* menghasilkan pesan *error* pada suatu perintah, pesan tersebut akan ditampilkan pada layer. Namun, hal ini tidak perlu dipermasalahkan. *Output* tidak di cetak pada layer melainkan akan menjadi *input* pada perintah selanjutnya. Karakter *pipe* sangat berguna, khususnya ketika ingin mencari data yang spesifik; karakter *pipe* sering digunakan untuk menyempurnakan hasil dari perintah awal.

Dalam contoh sebelumnya perintah `head` dan `tail` diberikan file sebagai argumen untuk mengoperasikannya. Namun, karakter *pipe* memungkinkan Anda untuk menggunakan perintah ini tidak hanya pada file, tetapi juga pada *output* dari perintah lain. Ini dapat berguna saat membuat daftar direktori besar, contohnya seperti direktori `/etc` :

```
sysadmin@localhost:~$ ls /etc
X11                     gss             mke2fs.conf     rpc
adduser.conf            host.conf       modprobe.d      rsyslog.conf
alternatives            hostname        modules         rsyslog.d
apparmor                hosts           modules-load.d  securetty
apparmor.d              hosts.allow     motd            security
apt                     hosts.deny      mtab            selinux
```

```
bash.bashrc                 init.d              nanorc              servic
es
bind                        initramfs-tools     netplan             shadow
bindresvport.blacklist      inputrc             network             shadow
-
binfmt.d                    insserv.conf.d      networks            shells
ca-certificates             iproute2            newt                skel
ca-certificates.conf        issue               nsswitch.conf       ssh
calendar                    issue.net           opt                 ssl
console-setup               kernel              os-release          subgid
cron.d                      ld.so.cache         pam.conf            subgid
-
cron.daily                  ld.so.conf          pam.d               subuid
cron.hourly                 ld.so.conf.d        passwd              subuid
-
cron.monthly                ldap                passwd-             sudoer
s
cron.weekly                 legal               perl                sudoer
s.d
crontab                     libaudit.conf       pinforc             sysctl
.conf
dbus-1                      locale.alias        ppp                 sysctl
.d
debconf.conf                locale.gen          profile             system
d
debian_version              localtime           profile.d           termin
fo
default                     logcheck            protocols           timezo
ne
deluser.conf                login.defs          python3             tmpfil
es.d
depmod.d                    logrotate.conf      python3.6           ucf.co
nf
dhcp                        logrotate.d         rc0.d               udev
dpkg                        lsb-release         rc1.d               ufw
environment                 machine-id          rc2.d               update
-motd.d
fstab                       magic               rc3.d               update
db.conf
gai.conf                    magic.mime          rc4.d               vim
```

```
groff                   mailcap          rc5.d           vtrgb
group                   mailcap.order    rc6.d           wgetrc
group-                  manpath.config   rcS.d           xdg
gshadow                 mc               resolv.conf
gshadow-                mime.types       rmt
```

Perintah sebelumnya mencantumkan banyak file. Jika Anda menjalankan ini di terminal netacad, *output* terputus dan hanya dapat dilihat jika di-scroll ke atas. Untuk lebih mudah melihat awal *output*, *pipe* ke perintah `head`. Contoh dibawah ini hanya menampilkan sepuluh baris pertama:

```
sysadmin@localhost:~$ ls /etc | head
X11
adduser.conf
alternatives
apparmor
apparmor.d
apt
bash.bashrc
bind
bindresvport.blacklist
binfmt.d
```

Output lengkap untuk perintah `ls` diletakkan di perintah `head` pada shell, bukan di cetak di layar. Perintah `head` membawa *output* dari perintah `ls` sebagai *input* data, dan *output* dari `head` dicetak ke layar.

Beberapa *pipe* dapat digunakan secara berurutan untuk menghubungkan beberapa perintah. Jika tiga perintah disatukan, *output* dari perintah pertama diteruskan ke perintah kedua. Kemudian, *output* dari perintah kedua diteruskan ke perintah ketiga. *Output* dari perintah ketiga kemudian akan dicetak ke layar.

Penting untuk hati-hati memilih urutan perintah yang disalurkan, karena setiap perintah hanya melihat *input* dari perintah sebelumnya. Contoh di bawah

mengilustrasikan penggunaan perintah `nl`, yang menambahkan nomor baris ke *output*. Pada contoh pertama, perintah `nl` digunakan untuk memberi nomor pada baris *output* dari perintah `ls` sebelumnya:

```
sysadmin@localhost:~$ ls /etc/ssh | nl
     1  moduli
     2  ssh_config
     3  ssh_host_ecdsa_key
     4  ssh_host_ecdsa_key.pub
     5  ssh_host_ed25519_key
     6  ssh_host_ed25519_key.pub
     7  ssh_host_rsa_key
     8  ssh_host_rsa_key.pub
     9  ssh_import_id
    10  sshd_config
```

Pada contoh berikutnya, perhatikan bahwa perintah `ls` dijalankan terlebih dahulu dan *outputnya* dikirim ke perintah `nl`, memberi nomor semua baris dari *output* perintah `ls`. Kemudian perintah `tail` dijalankan, menampilkan lima baris terakhir dari output perintah `nl`:

```
sysadmin@localhost:~$ ls /etc/ssh | nl | tail -5
     6  ssh_host_ed25519_key.pub
     7  ssh_host_rsa_key
     8  ssh_host_rsa_key.pub
     9  ssh_import_id
    10  sshd_config
```

Bandingkan *output* diatas dengan yang dibawah ini :

```
sysadmin@localhost:~$ ls /etc/ssh | tail -5 | nl
     1  ssh_host_ed25519_key.pub
     2  ssh_host_rsa_key
     3  ssh_host_rsa_key.pub
```

```
     4  ssh_import_id
     5  sshd_config
```

Terdapat perbedaan pada nomor baris. Hal ini di sebabkan :

Di contoh kedua, *output* perintah `ls` yang pertama di kirim ke perintah `tail` yang hanya membawa lima baris terakhir pada *output.* Lalu perintah `tail` mengirim lima baris ke perintah `nl`, yaitu nomor 1-5.

Karakter *pipe* dapat berguna, namun perlu mempertimbangkan bagaimana perintah tersebut disalurkan untuk memastikan bahwa *output* yang diinginkan ditampilkan.

3. INPUT/OUTPUT REDIRECTION

I/O Redirection memungkinkan informasi baris perintah untuk diteruskan ke aliran yang berbeda. Sebelum membahas pengalihan, penting untuk memahami tentang aliran standar.

- **STDIN**

  Standard Input, atau STDIN, adalah informasi yang dimasukkan secara normal oleh pengguna melalui keyboard. Ketika sebuah perintah meminta shell untuk data, shell memberi pengguna kemampuan untuk mengetik perintah untuk dikirim ke perintah sebagai STDIN secara bergilir.

- **STDOUT**

  Standard Output, atau STDOUT, adalah output normal dari perintah. Ketika sebuah perintah berfungsi dengan benar (tanpa kesalahan), keluaran yang dihasilkannya disebut STDOUT. Secara default, STDOUT ditampilkan di jendela terminal tempat perintah dijalankan. STDOUT juga dikenal sebagai *channel* atau saluran # 1.

- **STDERR**

Standard Error, atau STDERR, adalah pesan kesalahan yang dihasilkan oleh perintah. Secara default, STDERR ditampilkan di jendela terminal tempat perintah dijalankan. STDERR juga dikenal sebagai channel atau saluran # 2.

I/O Redirection memungkinkan pengguna untuk mengarahkan STDIN sehingga data berasal dari file dan STDOUT / STDERR sehingga output masuk ke file. Pengalihan dicapai dengan menggunakan karakter panah (<>).

3.1 STDOUT

STDOUT dapat diarahkan ke file. Untuk memulai, amati output dari perintah echo berikut yang ditampilkan ke layar:

```
sysadmin@localhost:~$ echo "Line 1"
Line 1
```

Menggunakan karakter (>), output dapat diarahkan ke file :

```
sysadmin@localhost:~$ echo "Line 1" > example.txt
```

Perintah ini tidak menampilkan *output* karena STDOUT dikirim ke file example.txt bukan ditampilkan pada layar. Anda dapat melihat file baru yaitu “example.txt” pada output dari perintah `ls`.

```
sysadmin@localhost:~$ ls
Desktop    Downloads  Pictures  Templates  example.txt
Documents  Music      Public    Videos
```

File tersebut berisi output dari perintah echo, yang dapat dilihat dengan perintah `cat`:

```
sysadmin@localhost:~$ cat example.txt
Line 1
```

Perlu diingat bahwa karakter *single arrow* menimpa konten apa pun dari konten file yang ada sebelumnya:

```
sysadmin@localhost:~$ cat example.txt
Line 1
sysadmin@localhost:~$ echo "New line 1" > example.txt
sysadmin@localhost:~$ cat example.txt
New line 1
```

Konten asli file hilang, diganti dengan output dari perintah echo baru.

Namun kita dapat mempertahankan isi konten file lama lalu menambah kan isi file dengan konten yang baru dengan menggunakan karakter dua panah (>>). Berikut contoh penggunaannya :

```
sysadmin@localhost:~$ cat example.txt
New line 1
sysadmin@localhost:~$ echo "Another line" >> example.txt
sysadmin@localhost:~$ cat example.txt
New line 1
Another line
```

Isi konten file yang lama tidak akan hilang, dan dibawah konten lama tersebut terdapat konten baru yaitu " Another Line" yang baru saja ditambahkan.

3.2 STDERR

STDERR dapat dialihkan juga mirip dengan STDOUT. Karakter (>) harus ditentukan agar kita dapat menganalisa apakah ini aliran STDOUT atau STDERR. Karna jika tidak ditentukan dengan angka 1 atau 2 sebelum karakter panah, maka computer akan membaca secara langsung bahwa aliran tersebut adalah STDOUT. Maka dari itu, angka 1 atau 2 harus ditentukan di awal. 2> untuk STDERR.

Untuk mendemonstrasikan pengalihan STDERR, pertama-tama amati perintah berikut yang menghasilkan kesalahan karena direktori yang ditentukan tidak ada:

```
sysadmin@localhost:~$ ls /fake
ls: cannot access /fake: No such file or directory
```

Perhatikan bahwa tidak ada dalam contoh di atas yang menyiratkan bahwa outputnya adalah STDERR. Keluarannya jelas merupakan pesan kesalahan, tetapi bagaimana Anda bisa tahu bahwa itu sedang dikirim ke STDERR? Salah satu cara mudah untuk menentukannya adalah dengan mengalihkan STDOUT:

```
sysadmin@localhost:~$ ls /fake > output.txt
ls: cannot access /fake: No such file or directory
```

Pada contoh di atas, STDOUT dialihkan ke file output.txt. Jadi, *output* yang ditampilkan tidak bisa STDOUT karena akan ditempatkan di file output.txt, bukan di terminal. Karena semua *output* perintah baik untuk STDOUT atau STDERR yang ditampilkan di atas harus STDERR.

Output STDERR dari sebuah perintah dapat dikirim ke file:

```
sysadmin@localhost:~$ ls /fake 2> error.txt
```

Dalam contoh, 2> menunjukkan bahwa semua pesan kesalahan harus dikirim ke file error.txt, yang dapat dikonfirmasi menggunakan perintah `cat`:

```
sysadmin@localhost:~$ cat error.txt
ls: cannot access /fake: No such file or directory
```

3.3 REDIRECTING MULTIPLE STREAMS

Dimungkinkan untuk mengarahkan STDOUT dan STDERR dari sebuah perintah pada saat yang bersamaan. Perintah berikut menghasilkan STDOUT dan STDERR karena salah satu direktori yang ditentukan ada dan yang lainnya tidak:

```
sysadmin@localhost:~$ ls /fake /etc/ppp
ls: cannot access /fake: No such file or directory
```

```
/etc/ppp:
ip-down.d  ip-up.d
```

Jika hanya STDOUT yang dikirim ke sebuah file, STDERR masih dicetak ke layar:

```
sysadmin@localhost:~$ ls /fake /etc/ppp > example.txt
ls: cannot access /fake: No such file or directory
sysadmin@localhost:~$ cat example.txt
/etc/ppp:
ip-down.d
ip-up.d
```

Jika hanya STDERR yang dikirim ke sebuah file, STDOUT masih dicetak ke layar:

```
sysadmin@localhost:~$ ls /fake /etc/ppp 2> error.txt
/etc/ppp:
ip-down.d
ip-up.d
sysadmin@localhost:~$ cat error.txt
ls: cannot access /fake: No such file or directory
```

Baik STDOUT dan STDERR dapat dikirim ke file dengan menggunakan karakter ampersand (&) di depan karakter panah (>). Jika terdapat karakter ampersand digabung dengan karakter panah (&>), maka itu berarti gabungan 1> dan 2>:

```
sysadmin@localhost:~$ ls /fake /etc/ppp &> all.txt
sysadmin@localhost:~$ cat all.txt
ls: cannot access /fake: No such file or directory
/etc/ppp:
ip-down.d
ip-up.d
```

Perhatikan bahwa saat Anda menggunakan karakter (&>), *output* akan muncul di file dengan semua pesan STDERR di atas dan semua pesan STDOUT di bawah semua pesan STDERR:

```
sysadmin@localhost:~$ ls /fake /etc/ppp /junk /etc/sound &> a
ll.txt
sysadmin@localhost:~$ cat all.txt
ls: cannot access '/fake': No such file or directory
ls: cannot access '/junk': No such file or directory
ls: cannot access '/etc/sound': No such file or directory
/etc/ppp:
ip-down.d
ip-up.d
```

Jika Anda tidak ingin STDERR dan STDOUT pergi ke file yang sama, mereka dapat dialihkan ke file yang berbeda dengan menggunakan karakter (>) dan 2>. Misalnya, untuk mengarahkan STDOUT ke example.txt dan STDERR ke error.txt, jalankan perintah berikut:

```
sysadmin@localhost:~$ ls /fake /etc/ppp > example.txt 2> erro
r.txt
sysadmin@localhost:~$ cat error.txt
ls: cannot access /fake: No such file or directory
sysadmin@localhost:~$ cat example.txt
/etc/ppp:
ip-down.d
ip-up.d
```

Urutan di mana aliran ditentukan tidak masalah.

3.4 STDIN

Konsep pengalihan STDIN adalah konsep yang sulit karena lebih sulit untuk memahami untuk mengalihkan STDIN. Dengan STDOUT dan STDERR, tujuan mereka langsung; terkadang sangat membantu untuk menyimpan *output* ke dalam file untuk digunakan di masa mendatang.

Sebagian besar pengguna Linux akhirnya mengarahkan STDOUT secara rutin, kadang-kadang STDERR, sedangkan sangat jarang untuk STDIN.

Ada sangat sedikit perintah yang mengharuskan Anda untuk mengarahkan STDIN karena dengan sebagian besar perintah jika ingin membaca data dari file ke dalam perintah, Anda dapat menentukan nama file sebagai argumen untuk perintah tersebut.

Untuk beberapa perintah, jika Anda tidak menentukan nama file sebagai argumen, perintah tersebut akan kembali menggunakan STDIN untuk mendapatkan data. Misalnya, perhatikan perintah `cat` berikut:

```
sysadmin@localhost:~$ cat
hello
hello
how are you?
how are you?
goodbye
goodbye
```

Catatan: Jika Anda mencoba perintah `cat` tanpa argumen, matikan proses dan kembali ke prompt dengan menggunakan Ctrl + C**.**

Dalam contoh sebelumnya, perintah cat tidak menyediakan nama file sebagai argumen. Jadi, ia meminta data untuk ditampilkan di layar dari STDIN. Pengguna mengetik ‘hello’, lalu perintah cat menampilkan ‘hello’ di layar. Perintah ini tidak begitu berguna.

Namun, jika *output* dari perintah cat diarahkan ke file, maka metode ini dapat digunakan untuk menambahkan teks ke file yang sudah ada atau untuk menempatkan teks ke file baru.

Perintah pertama pada contoh di bawah ini mengalihkan output dari perintah cat ke file yang baru dibuat bernama new.txt. Tindakan ini

ditindaklanjuti dengan memberikan perintah cat dengan file new.txt sebagai argumen untuk menampilkan teks yang diarahkan ulang di STDOUT.

```
sysadmin@localhost:~$ cat > new.txt
Hello
How are you?
Goodbye
sysadmin@localhost:~$ cat new.txt
Hello
How are you?
Goodbye
```

Sementara contoh sebelumnya menunjukkan keuntungan lain dari pengalihan STDOUT, contoh itu tidak membahas mengapa atau bagaimana STDIN dapat diarahkan. Untuk memahami ini, pertimbangkan perintah baru yang disebut `tr`. Perintah ini mengambil satu set karakter dan menerjemahkannya ke dalam set karakter lain.

Misalnya, untuk mengubah baris teks menjadi huruf besar, gunakan perintah tr sebagai berikut:

```
sysadmin@localhost:~$ tr 'a-z' 'A-Z'
watch how this works
WATCH HOW THIS WORKS
```

Perintah `tr` mengambil STDIN dari keyboard dan mengubah semua huruf kecil sebelum mengirim STDOUT ke layar.

Tampaknya penggunaan yang lebih baik dari perintah `tr` adalah melakukan terjemahan pada file, bukan input keyboard. Namun, perintah `tr` tidak mendukung argumen nama file:

```
sysadmin@localhost:~$ cat example.txt
/etc/ppp:
```

```
ip-down.d
ip-up.d
sysadmin@localhost:~$ tr 'a-z' 'A-Z' example.txt
tr: extra operand `example.txt'
Try `tr --help' for more information
```

Namun, dimungkinkan untuk memberi tahu shell untuk mendapatkan STDIN dari file daripada dari keyboard dengan menggunakan karakter (<):

```
sysadmin@localhost:~$ tr 'a-z' 'A-Z' < example.txt
/ETC/PPP:
IP-DOWN.D
IP-UP.D
```

Sebagian besar perintah menerima nama file sebagai argumen, jadi kasus penggunaan ini relatif jarang. Namun, bagi mereka yang tidak menerima nama file sebagai argumen, metode ini dapat digunakan agar shell membaca dari file daripada mengandalkan perintah untuk memiliki kemampuan ini.

Satu catatan terakhir untuk menyimpan *output* yang dihasilkan, alihkan ke file lain:

```
sysadmin@localhost:~$ tr 'a-z' 'A-Z' < example.txt > newexampl
e.txt
sysadmin@localhost:~$ cat newexample.txt
/ETC/PPP:
IP-DOWN.D
IP-UP.D
```

4. SORTING FILES

Perintah `sort` dapat digunakan untuk mengatur ulang baris file atau input baik dalam kamus atau urutan numerik. Contohnya membuat file kecil, menggunakan perintah `head` untuk mengambil 5 baris pertama dari file `/etc/passwd` dan mengirim output ke file bernama `mypasswd`.

```
sysadmin@localhost:~$ head -5 /etc/passwd > mypasswd
sysadmin@localhost:~$ cat mypasswd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
```

Sekarang kita akan mengurutkan (`sort`) file `mypasswd`:

```
sysadmin@localhost:~$ sort mypasswd
bin:x:2:2:bin:/bin:/usr/sbin/nologin
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
root:x:0:0:root:/root:/bin/bash
sync:x:4:65534:sync:/bin:/bin/sync
sys:x:3:3:sys:/dev:/usr/sbin/nologin
```

Setelah memeriksa dengan cermat output dalam contoh sebelumnya, perintah `sort` telah mengatur baris file dalam urutan abjad. Bandingkan output ini dengan output dari perintah `cat` sebelumnya.

4.1 OPSI FIELDS DAN SORT

Perintah sort dapat mengatur ulang output berdasarkan konten dari satu atau lebih kolom/*fields*. *Fields* ditentukan oleh *field delimiter* yang terdapat di setiap baris. Dalam komputasi, *delimeter* adalah karakter yang memisahkan string teks atau data; defaultnya adalah spasi/*whitespace*, seperti spasi/*space* atau tab.

Perintah berikut dapat digunakan untuk mengurutkan *field* ketiga dari file mypasswd secara numerik. Tiga opsi digunakan untuk memperoleh jenis ini:

Opsi	Fungsi

Opsi	Fungsi
`-t:`	Opsi `-t` menentukan pembatas bidang/*field delimeter*. Jika file atau input dipisahkan oleh pemisah/*delimeter* selain spasi/*whitespace*, misalnya koma atau titik dua, opsi `-t` akan memungkinkan pemisah bidang lain untuk ditetapkan sebagai argumen. File `mypasswd` yang digunakan dalam contoh sebelumnya menggunakan karakter titik dua : sebagai pemisah/*delimeter* untuk memisahkan *fields*, jadi contoh berikut menggunakan opsi `-t:` .
`-k3`	Opsi `-k` menentukan nomor bidang/*field number*. Untuk menentukan *field* mana yang akan diurutkan, gunakan opsi `-k` dengan argumen untuk menunjukkan nomor bidang/*field number*, dimulai dengan 1 untuk *field* pertama. Contoh berikut menggunakan opsi `-k3` untuk mengurutkan berdasarkan *field* ketiga.
`-n`	Opsi ini menentukan jenis pengurutan/*sort type*. Field ketiga di file `mypasswd` berisi angka, jadi opsi `-n` digunakan untuk melakukan pengurutan numerik.

```
sysadmin@localhost:~$ sort -t: -n -k3 mypasswd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
```

```
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
```

Opsi lain yang biasa digunakan untuk perintah `sort` adalah opsi `-r`, yang digunakan untuk melakukan pengurutan terbalik/*reverse sort*. Berikut ini adalah perintah yang sama seperti contoh sebelumnya, dengan tambahan opsi `-r`, membuat angka yang lebih tinggi di *field* ketiga yang muncul di bagian atas output:

```
sysadmin@localhost:~$ sort -t: -n -r -k3 mypasswd
sync:x:4:65534:sync:/bin:/bin/sync
sys:x:3:3:sys:/dev:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
root:x:0:0:root:/root:/bin/bash
```

Terakhir, Anda mungkin ingin melakukan pengurutan yang lebih kompleks, seperti mengurutkan menurut *field* utama lalu menurut *field* sekunder. Misalnya, pertimbangkan file nilai yang dipisahkan koma berikut, file dengan karakter koma sebagai pemisah bidang/*field delimeter*:

Catatan : Contoh ini untuk tujuan demonstrasi. File os.csv tidak ada di VM kami, oleh karena itu output mungkin berbeda di VM.

```
sysadmin@localhost:~$ cat os.csv
1970,Unix,Richie
1987,Minix,Tanenbaum
1970,Unix,Thompson
1991,Linux,Torvalds
```

Untuk mengurutkan terlebih dahulu berdasarkan sistem operasi (*field* # 2) dan kemudian tahun (*field* # 1) dan kemudian dengan nama belakang (*field* # 3), gunakan perintah berikut:

```
sysadmin@localhost:~$ sort -t, -k2 -k1n -k3 os.csv
1991,Linux,Torvalds
1987,Minix,Tanenbaum
1970,Unix,Richie
1970,Unix,Thompson
```

Tabel berikut membongkar opsi yang digunakan dalam contoh sebelumnya:

Opsi	Fungsi
`-t,`	Menentukan karakter koma sebagai pembatas bidang/*field delimiter*
`-k2`	Sortir berdasarka *field* # 2
`-k1n`	Mengurutkan secara *numerik* berdasarkan *field* # 1
`-k3`	Mengurutkan berdasarkan *field* # 3

5. STATISTIK FILE

Perintah `wc` menyediakan jumlah baris, kata dan byte (1 byte = 1 karakter dalam file teks) untuk sebuah file, dan jumlah baris total jika lebih dari satu file ditentukan. Secara default, perintah `wc` memungkinkan hingga tiga statistik

dicetak untuk setiap file yang disediakan, serta total statistik ini jika tersedia lebih dari satu nama file:

```
sysadmin@localhost:~$ wc /etc/passwd /etc/passwd-
  35   56 1710 /etc/passwd
  34   55 1665 /etc/passwd-
  69  111 3375 total
```

Output dari contoh sebelumnya memiliki empat kolom:

1. Jumlah baris
2. Jumlah kata
3. Jumlah byte
4. Nama file

Juga memungkinkan untuk melihat statistik tertentu saja, dengan menggunakan opsi `-l` untuk menunjukkan jumlah baris saja, opsi `-w` untuk menampilkan jumlah kata, opsi `-c` untuk menampilkan jumlah byte, atau kombinasi dari opsi ini.

Perintah `wc` juga berguna untuk menghitung jumlah baris yang dikeluarkan oleh beberapa perintah lain melalui *pipe*. Misalnya, jika Anda ingin mengetahui jumlah total file di direktori /etc, *pipe* output dari `ls` ke `wc` dan hitung hanya jumlah baris:

```
sysadmin@localhost:~$ ls /etc/ | wc -l
142
```

6. MEMFILTER FILE

Perintah `cut` dapat mengekstrak kolom teks dari file atau input standar. Digunakan terutama untuk bekerja dengan file database yang dibatasi. Sekali

lagi, file yang dipisahkan adalah file yang berisi kolom yang dipisahkan oleh pemisah/*delimiter*. File-file ini sangat umum di sistem Linux.

Secara default, perintah `cut` mengharapkan inputnya dipisahkan oleh karakter tab, tetapi opsi `-d` dapat menentukan pembatas alternative/*alternative delimeters* seperti titik dua atau koma.

Opsi `-f` dapat menentukan *field* mana yang akan ditampilkan, baik sebagai rentang yang diberi tanda hubung atau daftar yang dipisahkan koma.

Dalam contoh berikut, *field* pertama, kelima, keenam dan ketujuh dari file database mypasswd ditampilkan:

```
sysadmin@localhost:~$ cut -d: -f1,5-7 mypasswd
root:root:/root:/bin/bash
daemon:daemon:/usr/sbin:/usr/sbin/nologin
bin:bin:/bin:/usr/sbin/nologin
sys:sys:/dev:/usr/sbin/nologin
sync:sync:/bin:/bin/sync
```

Perintah `cut` juga dapat mengekstrak kolom teks berdasarkan posisi karakter dengan opsi `-c` — berguna saat bekerja dengan file database dengan lebar tetap atau keluaran perintah.

Misalnya, bidang perintah `ls -l` selalu dalam posisi karakter yang sama. Berikut ini hanya akan menampilkan jenis file (karakter 1), izin (karakter 2-10), spasi (karakter 11), dan nama file (karakter 50+):

```
sysadmin@localhost:~$ ls -l | cut -c1-11,50-
total 44
drwxr-xr-x Desktop
drwxr-xr-x Documents
drwxr-xr-x Downloads
drwxr-xr-x Music
drwxr-xr-x Pictures
```

```
drwxr-xr-x Public
drwxr-xr-x Templates
drwxr-xr-x Videos
-rw-rw-r-- all.txt
-rw-rw-r-- example.txt
-rw-rw-r-- mypasswd
-rw-rw-r-- new.txt
```

7. FILTER KONTEN PADA FILE

Perintah `grep` dapat digunakan untuk memfilter baris dalam file atau output dari perintah lain yang cocok dengan pola tertentu. Pola itu bisa sesederhana teks yang ingin Anda cocokkan atau bisa jauh lebih maju melalui penggunaan *regular expression*.

Misalnya, untuk menemukan semua pengguna yang dapat masuk ke sistem dengan shell BASH, perintah `grep` dapat digunakan untuk memfilter baris dari file `/etc/passwd` untuk baris yang berisi pola bash:

```
sysadmin@localhost:~$ grep bash /etc/passwd
root:x:0:0:root:/root:/bin/bash
sysadmin:x:1001:1001:System Administrator,,,,:/home/sysadmin
:/bin/bash
```

Untuk mempermudah melihat apa yang benar-benar cocok, gunakan opsi `--color`. Opsi ini akan menyorot item yang cocok dengan warna merah:

```
sysadmin@localhost:~$ grep --color bash /etc/passwd
root:x:0:0:root:/root:/bin/bash
sysadmin:x:1001:1001:System Administrator,,,,:/home/sysadmin
:/bin/bash
```

Catatan: Di mesin virtual kami, perintah grep dialias untuk menyertakan opsi --color secara otomatis.

Dalam beberapa kasus, mungkin tidak penting untuk menemukan garis tertentu yang cocok dengan pola, melainkan berapa banyak garis yang cocok dengan pola tersebut. Opsi `-c` memberikan hitungan berapa banyak baris yang cocok:

```
sysadmin@localhost:~$ grep -c bash /etc/passwd
2
```

Saat melihat output dari perintah `grep`, mungkin sulit untuk menentukan nomor baris aslinya. Informasi ini dapat berguna saat kembali ke file (mungkin untuk mengedit file) untuk menemukan salah satu baris yang cocok dengan cepat.

Opsi `-n` pada perintah `grep` akan menampilkan nomor baris asli. Untuk menampilkan semua baris dan nomor barisnya di file `/etc/passwd` yang berisi pola `bash`:

```
sysadmin@localhost:~$ grep -n bash /etc/passwd
1:root:x:0:0:root:/root:/bin/bash
27:sysadmin:x:1001:1001:System Administrator,,,,:/home/sysad
min:/bin/bash
```

Opsi `-v` membalikkan kecocokan, mengeluarkan semua baris yang tidak mengandung pola. Untuk menampilkan semua baris yang tidak mengandung `nologin` di file `/etc/passwd`:

```
sysadmin@localhost:~$ grep -v nologin /etc/passwd
root:x:0:0:root:/root:/bin/bash
sync:x:4:65534:sync:/bin:/bin/sync
operator:x:1000:37::/root:/bin/sh
sysadmin:x:1001:1001:System Administrator,,,,:/home/sysadmin
:/bin/bash
```

Opsi `-i` mengabaikan perbedaan huruf besar(kapital). Pencarian berikut untuk pola the di `newhome.txt`, memungkinkan setiap karakter menjadi huruf besar atau kecil:

```
sysadmin@localhost:~$ cd Documents
sysadmin@localhost:~/Documents$ grep -i the newhome.txt
There are three bathrooms.
**Beware** of the ghost in the bedroom.
The kitchen is open for entertaining.
**Caution** the spirits don't like guests.
```

Opsi `-w` hanya mengembalikan baris yang berisi kecocokan yang membentuk seluruh kata. Untuk menjadi sebuah kata, string karakter harus diawali dan diikuti dengan karakter non-kata. Karakter kata termasuk huruf, angka, dan karakter garis bawah.

Contoh berikut mencari pola are di file `newhome.txt`. Perintah pertama mencari tanpa opsi, sedangkan perintah kedua menyertakan opsi `-w`. Bandingkan hasilnya:

```
sysadmin@localhost:~/Documents$ grep are newhome.txt
There are three bathrooms.
**Beware** of the ghost in the bedroom.
sysadmin@localhost:~/Documents$ grep -w are newhome.txt
There are three bathrooms.
```

8. BASIC REGULAR EXPRESSION

Regular expressions, juga disebut *regex*, adalah kumpulan karakter normal dan khusus yang masing-masing digunakan untuk menemukan pola sederhana atau kompleks dalam file. Karakter ini digunakan untuk melakukan fungsi pencocokan tertentu dalam pencarian.

Karakter normal adalah karakter alfanumerik yang cocok dengan dirinya sendiri. Misalnya, a akan cocok dengan a. Karakter khusus memiliki arti khusus saat digunakan dalam pola dengan perintah seperti perintah grep. Mereka berperilaku lebih kompleks dan tidak cocok dengan dirinya sendiri.

Ada *Basic Regular Expressions* (tersedia untuk berbagai macam perintah Linux) dan *Extended Regular Expressions* (tersedia untuk perintah Linux yang lebih canggih). Basic Regular Expressions meliputi:

Karakter	Fungsi
.	Setiap karakter tunggal
[ ]	Daftar atau rentang karakter yang cocok dengan satu karakter. Jika karakter pertama adalah tanda caret ^, itu berarti karakter tidak ada dalam daftar.
*	Karakter sebelumnya mengulangi nol atau lebih banyak lagi.
^	Jika karakter pertama dalam pola, polanya harus di awal baris untuk mencocokkan, jika tidak hanya literal ^
$	Jika karakter terakhir dalam pola, polanya harus di akhir baris untuk mencocokkan, jika tidak hanya literal $

Perintah grep hanyalah salah satu dari banyak perintah yang mendukung *Regular expressions*. Beberapa perintah termasuk more dan less.

Meskipun beberapa ekspresi reguler tidak perlu dikutip dengan tanda kutip tunggal, praktik yang baik adalah menggunakan tanda kutip tunggal di sekitar ekspresi reguler untuk mencegah shell mencoba menafsirkan makna khusus darinya.

8.1 KARAKTER DOT (.)

Salah satu ekspresi yang paling berguna adalah dot (.). karakter Ini cocok dengan karakter apa pun kecuali untuk karakter baris baru. Berikut merupakan file `~/Documents/red.txt` yang belum terfilter oleh karakter dot (.) :

```
sysadmin@localhost:~/Documents$ cat red.txt
red
```

```
reef
rot
reeed
rd
rod
roof
reed
root
reel
read
```

Pola `r..f` akan menemukan banyak baris yang mengandung huruf `'r'` yang diikuti oleh 2 karakter dan lalu diikuti oleh huruf `'f'`

```
sysadmin@localhost:~/Documents$ grep 'r..f' red.txt
reef
roof
```

Baris tersebut tidak harus sama persis, tetapi harus mengandung pola, seperti yang terlihat di sini saat `r..t` dicari di file `/ etc / passwd`:

```
sysadmin@localhost:~/Documents$ grep 'r..t' /etc/passwd
root:x:0:0:root:/root:/bin/bash
operator:x:1000:37::/root:
```

Karakter dot (.) dapat digunakan dalam jumlah berapapun. Untuk menemukan seluruh kata yang setidaknya memiliki 4 karakter, pola berikut ini dapat digunakan :

```
sysadmin@localhost:~/Documents$ grep '....' red.txt
reef
reeed
roof
reed
root
```

```
reel
read
```

8.2 KARAKTER SQUARE BRACKET ([])

Ketika menggunakan karakter dot (.), karakter apa pun bisa cocok dengannya. Dalam beberapa kasus, Anda ingin menentukan dengan tepat karakter yang ingin Anda cocokkan, seperti karakter alfabet huruf kecil atau karakter angka.

Tanda *square bracket* ([]) dipergunakan untuk menemukan suatu karakter dari daftar atau rentang karakter yang mungkin karakter tersebut terkandung di dalam tanda kurung. Misalnya, dicontohkan dalam file `profile.txt`:

```
sysadmin@localhost:~/Documents$ cat profile.txt
Hello my name is Joe.
I am 37 years old.
3121991
My favorite food is avocados.
I have 2 dogs.
123456789101112
```

Untuk menemukan baris di dalam file `profile.txt` yang mana memiliki karakter nomor di dalamnya, gunakan pola `[0123456789]` atau `[0-9]`:

```
sysadmin@localhost:~/Documents$ grep '[0-9]' profile.txt
I am 37 years old.
3121991
I have 2 dogs.
123456789101112
```

Perhatikan bahwa pencarian karakter dengan *square brackets* ([]) dapat ditulis [abcd] atau di tulis sebagai rentang tertentu seperti [a-d], selama ditulis

dengan urutan yang benar, maka perintah yang dituliskan dapat menghasilkan output yang sesuai. Sebagai contoh, [d-a] tidak akan berfungsi karna itu adalah rentang yang tidak valid :

```
sysadmin@localhost:~/Documents$ grep '[d-a]' profile.txt
grep: Invalid range end
```

Rentang dispesifikasikan sesuai dengan standar table ASCII. Tabel ini terdapat kumpulan karakter yang dicetak dalam urutan tertentu. Anda dapat melihat table ASCII yang disertai perintah ASCII. Sebagai contoh :

```
041  33  21  !                    141  97  61  a
042  34  22  "                    142  98  62  b
043  35  23  #                    143  99  63  c
044  36  24  $                    144  100 64  d
045  37  25  %                    145  101 65  e
046  38  26  &                    146  102 66  f
```

Nilai numerik dari a adalah 141 dan nilai numerik d adalah 144. Nilai numerik a lebih kecil dari nilai numerik d; oleh karena itu rentang a-d adalah rentang yang valid.

Bagaimana dengan karakter selain numerik? Misalnya, untuk mencocokkan karakter yang bisa apa saja kecuali x, y atau z? Tidak efisien untuk menyediakan himpunan semua karakter kecuali x, y atau z.

Untuk mencocokkan karakter yang mana karakter tersebut bukan salah satu karakter yang terdaftar dalam range, sertakan simbol caret (^). Simbol tersebut untuk menemukan semua baris yang berisi karakter non-numerik, masukkan ^ sebagai karakter pertama di dalam tanda kurung. Karakter ini meniadakan karakter yang terdaftar:

```
sysadmin@localhost:~/Documents$ grep '[^0-9]' profile.txt
Hello my name is Joe.
```

```
I am 37 years old.
My favorite food is avocados.
I have 2 dogs.
```

Pertimbangkan Hal Ini :

Jangan keliru, rentang `[^0-9]` yang mana rentang tersebut disertai karakter caret (^) didepannya dipergunakan untuk mencocokan baris yang mana baris tersebut tidak terdapat angka. Rentang ini berfungsi untuk mencari nilai yang tidak mengandung angka. Pada file yang isinya terdapat baris yang hanya mengandung angka, tidak akan ditampilkan apabila mengetik rentang `[^0-9]`.

8.3 KARAKTER ASTERISK (*)

Karakter arterisk (*) digunakan untuk mencocokan 0 atau lebih karakter. Pencocokan terjadi antara range yang ditulis dalam perintah atau tanpa range tertentu dengan kalimat yang terdapat dalam suatu file. Sebagai contoh, `e*d` akan menghasilkan kalimat yang dimana kalimat tersebut terdapat huruf `e` dan `d`:

```
sysadmin@localhost:~/Documents$ cat red.txt
red
reef
rot
reeed
rd
rod
roof
reed
root
reel
read
sysadmin@localhost:~/Documents$ grep 're*d' red.txt
red
```

```
reeed
rd
reed
```

Dimungkinkan juga untuk mencocokkan nol atau lebih kemunculan daftar karakter dengan menggunakan tanda kurung siku. Pola [oe] * yang digunakan dalam contoh dibawah akan menampilkan output kalimat dalam file red.txt yang mengandung karakter o atau karakter e:

```
sysadmin@localhost:~/Documents$ grep 'r[oe]*d' red.txt
red
reeed
rd
rod
reed
```

Ketika digunakan hanya dengan 1 karakter, karakter * tidak dapat membantu. Salah satu pola berikut akan cocok dengan setiap string atau baris dalam file : '. *' 'E *' 'b *' 'z *' karena karakter asterisk (*) bisa cocok dengan nol kemunculan pola.

```
sysadmin@localhost:~/Documents$ grep 'z*' red.txt
red
reef
rot
reeed
rd
rod
roof
reed
root
reel
read
sysadmin@localhost:~/Documents$ grep 'e*' red.txt
```

```
red
reef
rot
reeed
rd
rod
roof
reed
root
reel
read
```

Untuk membuat karakter arterisk (*) berfungsi, perlu untuk membuat pola yang mana lebih dari 1 karakter sebelum karakter arterisk (*). Sebagai contoh, beberapa baris diatas dapat dihilangkan dengan menambahkan `e` lainnya untuk memuat pola `ee*` hasil menjadi lebih efektif dan sesuai di setiap baris yang mengandung setidaknya satu `e`.

```
sysadmin@localhost:~/Documents$ grep 'ee*' red.txt
red
reef
reeed
reed
reel
read
```

8.4 KARAKTER ANCHOR

Ketika menunjukan kesesuaian pola, kesesuaian dapat muncul dimanapun dalam baris. *Anchor characters* adalah satu dari beberapa *regular expression* yang dapat berfungsi mempersempit hasil pencarian. Mereka menentukan kesesuaian terdapat di awal atau akhir baris.

Sebagai contoh, pola `root` muncul beberapa kali di file `/etc/passwd` :

```
sysadmin@localhost:~/Documents$ grep 'root' /etc/passwd
root:x:0:0:root:/root:/bin/bash
operator:x:1000:37::/root:
```

Karakter caret (circumflex) (^) digunakan untuk memastikan bahwa pola muncul di awal baris. Sebagai contoh, untuk menemukan semua baris pada `/etc/passwd` yang diawali dengan ketikkan `root` gunakan pola `^root`. Pastikan karakter ^ harus berada di awal kata `root` agar tidak muncul banyak kata root :

```
sysadmin@localhost:~/Documents$ grep '^root' /etc/passwd
root:x:0:0:root:/root:/bin/bash
```

Karakter anchor yang kedua adalah *dollar sign* ($), yang digunakan untuk memastikan pola muncul di akhir baris, sehingga secara efektif mengurangi hasil pencarian. Untuk menemukan baris yang diakhiri dengan r di file `first.txt`, gunakan pola `r$`:

```
sysadmin@localhost:~/Documents$ cat alpha-first.txt
A is for Animal
B is for Bear
C is for Cat
D is for Dog
E is for Elephant
F is for Flower
sysadmin@localhost:~/Documents$ grep 'r$' alpha-first.txt
B is for Bear
F is for Flower
```

Lalu, posisi suatu karakter sangat penting. Karakter $ harus ada di posisi terakhir dalam suatu pola agar menjadi efektif dan hasil yang di dapat sesuai yang diharapkan.

8.5 KARAKTER BACKSLASH (\)

Dalam beberapa kasus, Anda mungkin ingin mencocokkan karakter yang kebetulan merupakan karakter ekspresi reguler khusus. Misalnya, perhatikan hal berikut :

```
sysadmin@localhost:~/Documents$ cat newhome.txt
Thanks for purchasing your new home!!

**Warning** it may be haunted.

There are three bathrooms.

**Beware** of the ghost in the bedroom.

The kitchen is open for entertaining.

**Caution** the spirits don't like guests.

Good luck!!!
sysadmin@localhost:~/Documents$ grep 're*' newhome.txt
Thanks for purchasing your new home!!
**Warning** it may be haunted.
There are three bathrooms.
**Beware** of the ghost in the bedroom.
The kitchen is open for entertaining.
**Caution** the spirits don't like guests.
```

Output dari perintah `grep` diatas, pencarian untuk `re*` sesuai setiap baris yang mana mengandung `r` diikuti oleh `0` atau lebih huruf `e`. Untuk mencari karakter asterisk (*) yang sebenarnya, letakkan karakter backslash (\) sebelum karakter asterisk (*):

```
sysadmin@localhost:~/Documents$ grep 're\*' newhome.txt
**Beware** of the ghost in the bedroom.
```

8.6 EXTENDED REGULAR EXPRESSION

Penggunaan *Extended Regular Expression* seringkali membutuhkan opsi khusus yang diberikan kepada perintah untuk mengenalinya. Menurut sejarah, ada perintah yang disebut `egrep`, yang sama dengan `grep`, tetapi dapat memahami *Extended Regular Expression*. Sekarang, perintah `egrep` tidak digunakan lagi karena menggunakan `grep` dengan opsi `-E` .

Karakter Extended Regular Expression :

Character	Matches
?	Cocokkan dengan karakter sebelumnya nol atau satu kali, jadi karakter ini adalah karakter opsional
+	Cocokkan dengan karakter sebelumnya yang diulang satu kali atau lebih.
|	Alternatif atau seperti operator *logical*.

Untuk mencari `colo` yang diikuti oleh nol atau satu huruf `u` yang diikuti oleh huruf `r` :

```
sysadmin@localhost:~/Documents$ grep -E 'colou?r' spelling.txt
American English: Do you consider gray to be a color or a shade?
British English: Do you consider grey to be a colour or a shade?
```

Untuk mencari satu atau lebih huruf `e`:

```
sysadmin@localhost:~/Documents$ grep -E 'e+' red.txt
red
reef
reeed
reed
reel
```

```
read
```

Untuk mencari `gray` atau `grey` :

```
sysadmin@localhost:~/Documents$ grep -E 'gray|grey' spelling.txt
American English: Do you consider gray to be a color or a shade?
British English: Do you consider grey to be a colour or a shade?
```

RANGKUMAN

1. Sebagian besar File Sistem adalah bertipe File Teks. Untuk mengelola file-file tersebut, kita memerlukan *command* atau perintah. Ada banyak sekali perintah pada linux, yang mana perintah-perintah tersebut mempunyai fungsinya masing-masing. Beberapa dari perintah tersebut adalah `Cat`, `Less`, `More`, `Head`, dan `Tail`. Penggunaan beberapa perintah tersebut tergantung dari besar kecilnya ukuran data pada file.
2. Input Output Redirection merupakan teknik untuk mengalihkan data ke salah satu streams yang dituju (STDIN / STDOUT / STDERR). Yang mana streams ini salah satu teknik mengelola data pada file. Teknik Redirection menggunakan karakter chevron arrow (<, >, 2>).
3. Setelah melakukan modifikasi atau penambahan data suatu file. Kita dapat menampilkannya pada terminal menggunakan beberapa perintah seperti pada poin pertama. Namun jika kita ingin memfilter bentuk tampilannya terlebih dahulu, kita dapat menggunakan perintah seperti `sort`, `cut`, dan `grep`.
4. Basic Regular Expression dan Extended Regular Expression merupakan kumpulan karakter normal dan spesial yang digunakan untuk menemukan pola string pada suatu file. Regular Expression mempermudah kita untuk menampilkan data yang kita butuhkan. Bedanya dengan perintah `cut` dan `sort` terletak pada bentuk tampilannya.

```
sysadmin@localhost:~$ cut -d: -f1,5-7 mypasswd
root:root:/root:/bin/bash
daemon:daemon:/usr/sbin:/usr/sbin/nologin
bin:bin:/bin:/usr/sbin/nologin
sys:sys:/dev:/usr/sbin/nologin
sync:sync:/bin:/bin/sync
```

Pada perintah `cut`, kita akan menampilkan fields yang diinginkan.

```
sysadmin@localhost:~/Documents$ grep '[0-9]' profile.txt
I am 37 years old.
3121991
I have 2 dogs.
123456789101112
```

Sedangkan dengan menggunakan perintah `grep` yang disertai pola Regular Expression, output akan menyeleksi konten sesuai dengan perintah yang diberikan oleh user.

Referensi:

https://www.netacad.com/courses/os-it/ndg-linux-essentials