**Laporan Latihan Praktikum ke-V  
Sistem Operasi**

**Disusun oleh:**

**RA Siti Zakiyah | 121140103**

**Kelas RD | Cluster RD**



**Program Studi Teknik Informatika**

**Jurusan Teknologi Produksi dan Industri**

**Institut Teknologi Sumatera**

**Lampung Selatan**

**2023**

**BAB I  
TEORI DASAR**

**1.1 Proses Input / Output**

Proses Input/Output (I/O) adalah cara komunikasi program dan perangkat input/output. Proses input meliputi perolehan data atau informasi dari sumber eksternal, seperti file, keyboard, atau mouse. Proses output melibatkan tampilan data atau informasi ke perangkat eksternal seperti monitor, printer, atau file. Proses input dilakukan melewati berbagai perangkat input, seperti keyboard atau mouse. Proses input juga dapat dilakukan melalui membaca data dari file, membaca input dari baris perintah atau melalui interaksi dengan user antarmuka grafis. Proses output melibatkan tampilan data kepada user atau menyimpan data ke tempat penyimpanan. Contoh tempat penyimpanan yaitu file. Output dapat berupa grafik, teks, atau suara tergantung jenis perangkat yang digunakan.

**1.2 File Descriptor**

File descriptor merupakan teknik yang digunakan oleh sistem operasi untuk mengidentifikasi dan mengelola file yang sedang dibuka oleh sebuah proses. Setiap file yang dibuka memiliki file descriptor yang unik yang biasanya berupa bilangan bulat non negatif. Sistem operasi menyediakan file descriptor sebagai abstraksi untuk mempresentasikan file yang dibuka oleh sebuah proses. File descriptor digunakan untuk melakukan operasi I/O pada file, seperti menulis membaca membuka dan menutup file. Terdapat tiga file deskriptor standar secara otomatis yang disediakan untuk setiap proses, yaitu file descriptor 0 (stdin) yang merupakan file descriptor standar untuk input yang biasanya terhubung ke keyboard atau sumber input lainnya, kemudian file descriptor 1 (stdout) merupakan file descriptor standar untuk output yang biasanya terhubung ke layar atau perangkat output lainnya dan file descriptor 2 (stderr yang merupakan file descriptor standar untuk output error yang biasanya terhubung ke layar atau perangkat output error lainnya.

**1.3 Redirection (Pembelokan)**

Redirection merupakan proses pengalihan input atau output dari atau ke sumber atau tujuan yang berbeda. Kita dapat menggunakan operator redirection untuk mengarahkan input atau output. Operator "<" digunakan untuk mengalihkan input dari file. Contohnya, perintah sort < input.txt akan mengambil input dari file "input.txt" daripada dari keyboard. Operator ">" digunakan untuk mengalihkan output ke file. Contohnya, perintah ls > file.txt akan mengalihkan output dari perintah ls ke file bernama "file.txt" daripada menampilkannya di layar. Operator ">>" digunakan untuk menambahkan output ke file yang sudah ada. Contohnya, perintah echo "baris baru" >> file.txt akan menambahkan "baris baru" ke dalam file "file.txt" tanpa menghapus kontennya yang sudah ada.

**1.4 Pipeline**

Pipeline adalah mekanisme yang digunakan untuk mengalirkan output suatu perintah atau program ke input perintah atau program lainnya dalam urutan yang terhubung. Output dari satu proses menjadi input untuk proses berikutnya, dan demikian seterusnya. Operator "|" (pipe) digunakan untuk menghubungkan sebuah perintah atau program dalam pipeline. Penggunaan pipeline memungkinkan user untuk melakukan pengolahan data yang kompleks dengan menggabungkan beberapa perintah atau program ke dalam satu baris. Ini memberikan fleksibilitas dan kekuatan dalam memanipulasi, menyaring, dan menganalisis data.

**1.5 Filter**

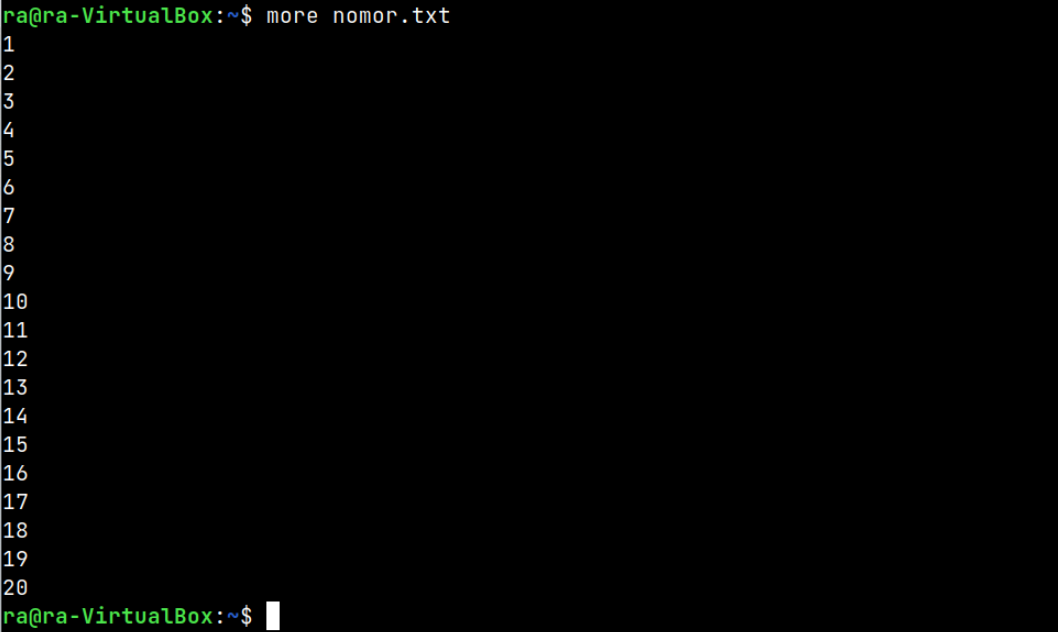
Filter merupakan program yang melakukan transformasi pada input yang diberikan dan menghasilkan output yang dimodifikasi. Filter biasanya digunakan dalam kombinasi dengan pipeline untuk melakukan manipulasi data secara berurutan. Filter dapat digunakan untuk mencocokkan pola teks, mengganti teks, mengurutkan data, menghitung statistik, dan melakukan berbagai operasi pemrosesan data lainnya. Beberapa contoh filter yang umum digunakan adalah "grep" untuk mencocokkan pola teks, "sort" untuk mengurutkan data, "sed" untuk melakukan penggantian teks, dan "awk" untuk melakukan pemrosesan teks berbasis pola. Dalam penggunaan filter, output dari satu filter digunakan sebagai input untuk filter berikutnya dalam pipeline. Dengan memadukan beberapa filter, pengguna dapat melakukan manipulasi dan analisis data secara efisien, memisahkan tugas ke dalam unit yang lebih kecil, dan memperoleh hasil yang diinginkan dengan langkah-langkah yang terurut.

**BAB II  
PEMBAHASAN DAN ANALISIS**

**2.1 Bagaimana perbedaan antara command more dan less, dan implementasikan pada terminal**

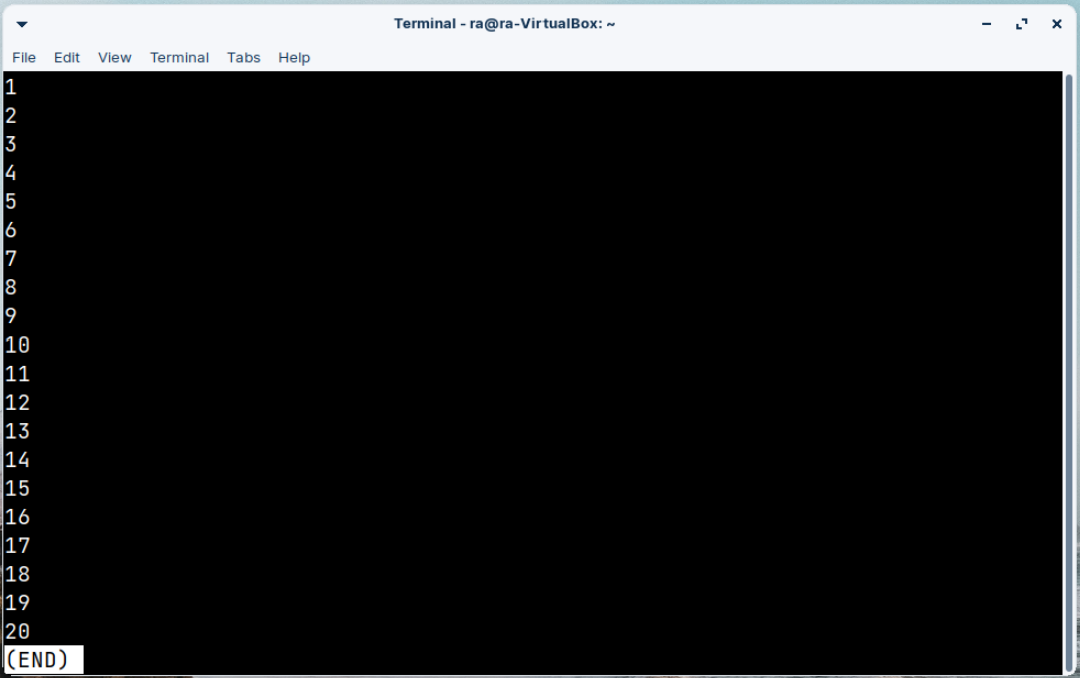
**2.1.1 More**

Sebelumnya kita membuat file huruf.txt yang berisi huruf A-Z. Kemudian melakukan percobaan **more** pada file huruf.txt dengan sintaks **more nomor.txt.** Dilihat bahwa command **more** dapat mengouputkan isi dari suatu file secara menyeluruh dalam terminal.

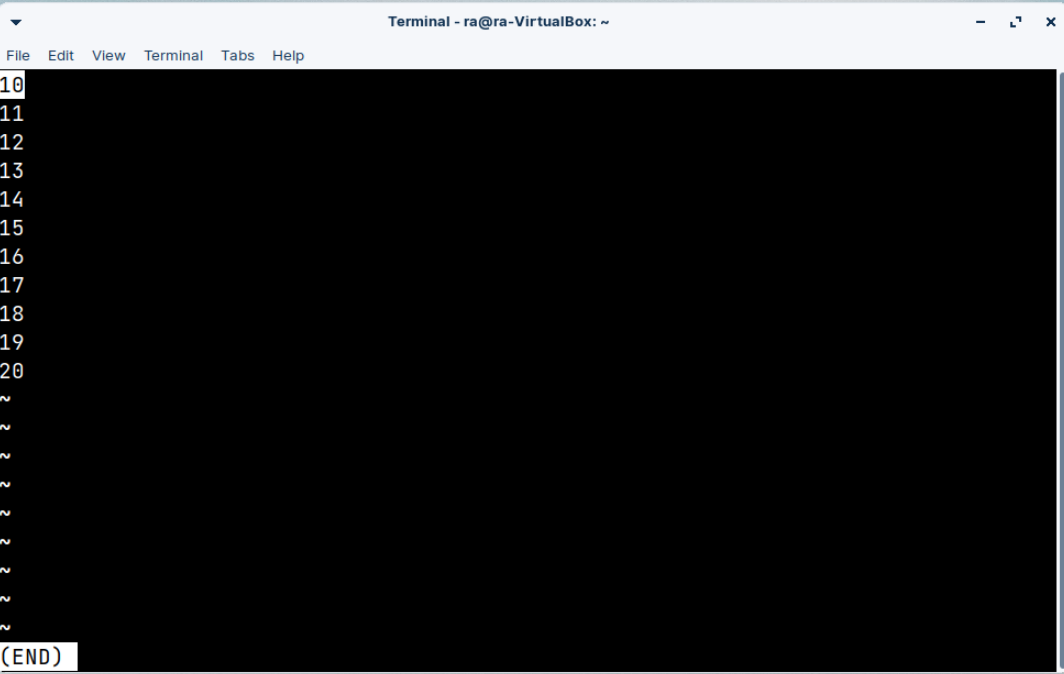


**2.1.2** **Less**

Untuk melakukan percobaan **less** pada file “nomor.txt” dapat menggunakan sintaks **less nomor.txt.** Dilihat bahwa command **less** dapat menampilkan isi dari suatu file secara menyeluruh, namun bukan dalam terminal yang sama saat menjalankan commandnya, melainkan terbuka terminal lain khusus command **less**.

****

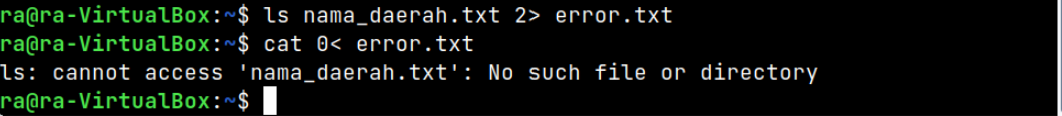
Kemudian command memiliki fitur untuk mencari suatu string yang terdapat dalam file tersebut. Contoh penggunaannya adalah “/10” yang berarti mencari string “10”, maka setelah mengklik **enter,** maka semua string yang mengandung nilai “10” akan terhighlight. Kemudian pada baris paling akhir terdapat text (END) yang menandakan akhir dari baris string yang dimiliki yang dimiliki file tersebut.

****

**2.2 Buatlah sebuah file yang berisikan daftar nama daerah dengan menggunakan file descriptor dan redirection dengan mengimplementasikan penggunaan standar input, standar output dan standar error.**

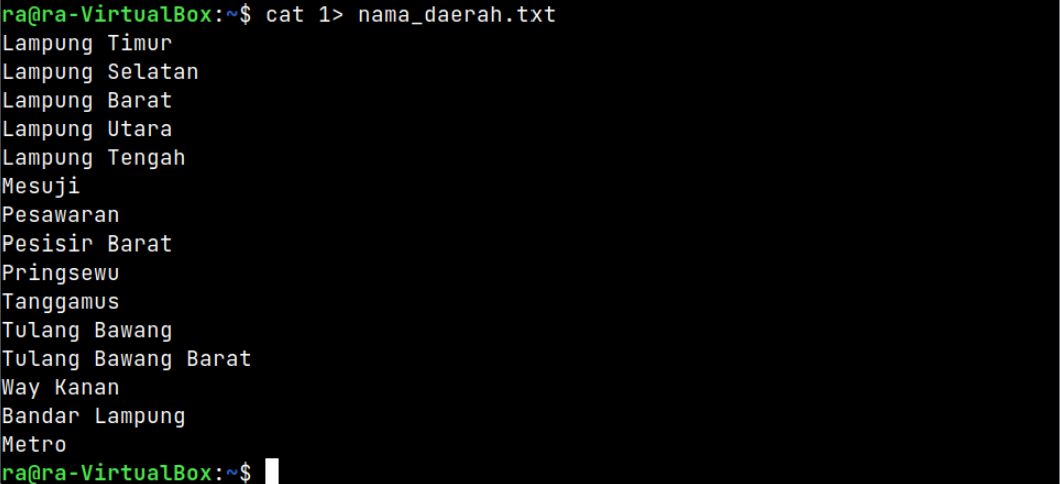
**2.2.1 Redirection standar error**

Untuk melakukan penerapan “direction standar error” pada list file dapat menggunakan perintah **ls** dengan sintaks **ls nama\_daerah.txt 2> error.txt**, maka file “error.txt” akan terisi hasil error yaitu **No such file or directory** dikarenakan file “nama\_daerah.txt” tidak ada atau belum dibuat.

****

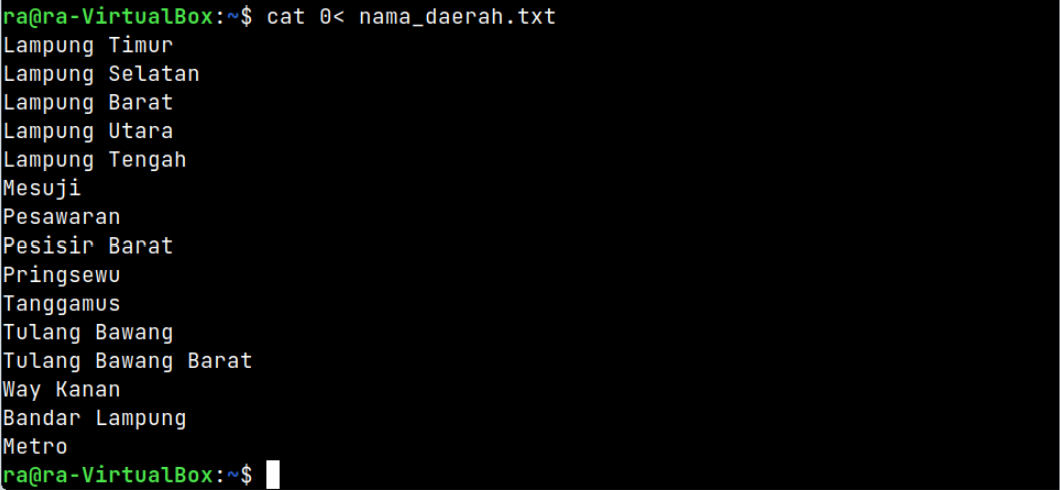
**2.2.2 Redirection standar output**

Untuk melakukan penerapan “redirection standar output” ke dalam file baru “nama\_daerah.txt” menggunakan sintaks **cat 1> nama\_daerah.txt**, kemudian menginputkan 15 nama kabupaten di Lampung, setelah itu, klik **ctrl+d** untuk mengakhiri inputan dan file akan otomatis tersimpan.

****

**2.2.3 Redirection standar input**

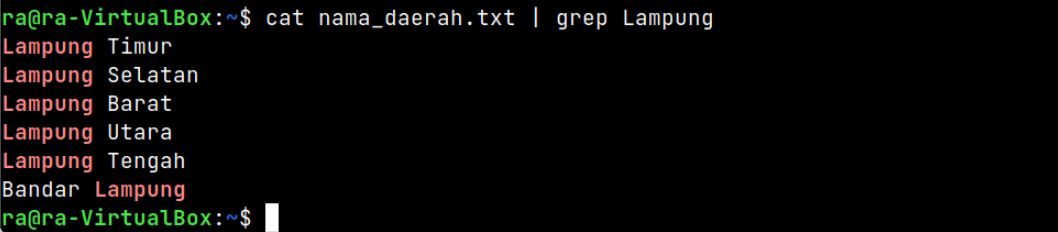
Untuk melakukan penerapan “redirection standar input” pada file “nama\_daerah.txt” dapat menggunakan sintaks **cat 0< nama\_daerah.txt**, kemudian akan keluar output isi file “nama\_daerah.txt” yang sebelumnya telah buat.

****

**2.3 Tampilkan isi file tersebut dengan menggunakan implementasi dari pipeline dan filter yaitu perintah Grep, Wc, Uniq, Sort, Head, Tail, Cut, Paste, Awk, Sed**

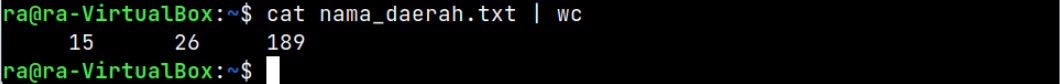
**2.3.1 Grep**

Untuk melakukan pipeline dan filter **Grep** pada file “nama\_daerah.txt” dapat menggunakan sintaks **cat nama\_daerah.txt | grep Lampung** untuk menampilkan baris string yang mengandung string “Lampung” saja.

****

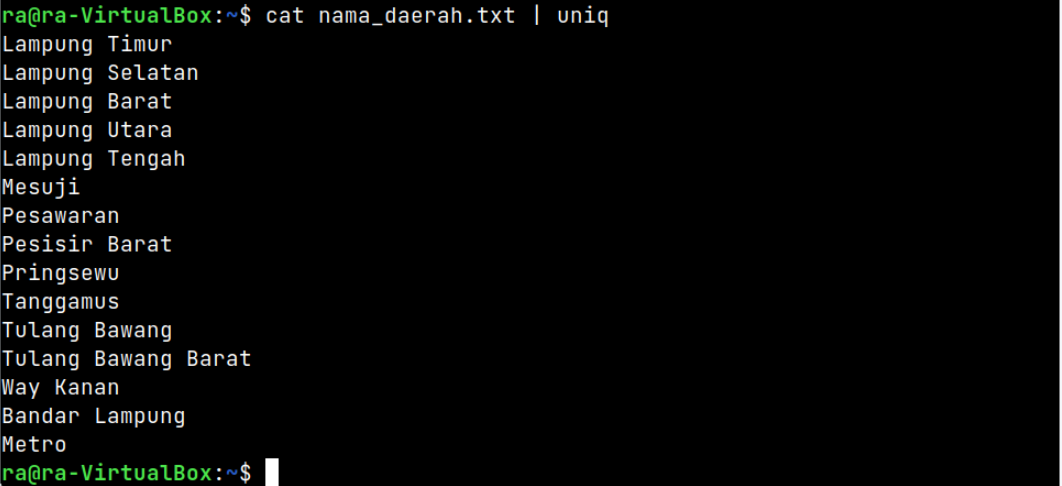
**2.3.2 Wc**

Untuk melakukan pipeline dan filter **Wc** pada file “nama\_daerah.txt” dapat menggunakan sintaks **cat nama\_daerah.txt | wc** untuk menampilkan jumlah baris, jumlah kata, dan jumlah karakter yang terdapat pada file.

****

**2.3.3 Uniq**

Untuk melakukan pipeline dan filter **Uniq** pada file “nama\_daerah.txt” dapat menggunakan sintaks **cat nama\_daerah.txt | uniq** untuk tidak menampilkan baris duplikasi, namun dalam file “nama\_daerah.txt” ini tidak ada baris duplikasi sehingga output filenya sama seperti biasa.

****

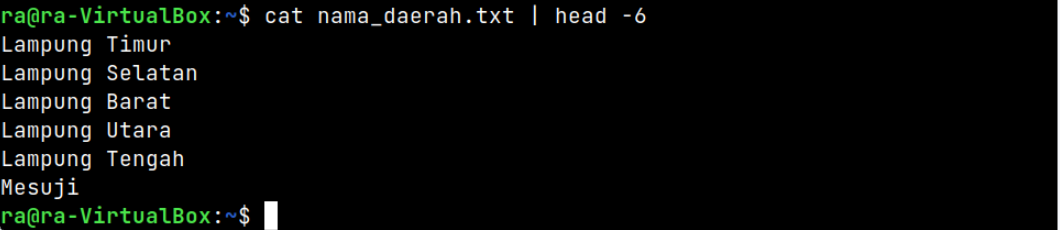
**2.3.4 Sort**

Untuk melakukan pipeline dan filter **Sort** pada file “nama\_daerah.txt” menggunakan sintaks **cat nama\_daerah.txt | sort** untuk mengurutkan baris sesuai dengan abjad menurun dari A sampai Z.

****

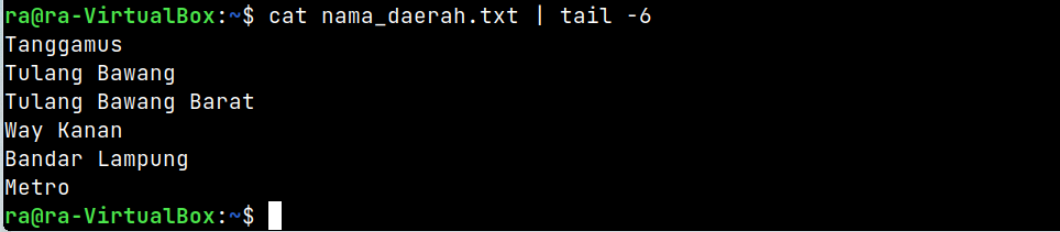
**2.3.5 Head**

Untuk melakukan pipeline dan filter **Head** pada file “nama\_daerah.txt” dapat menggunakan sintaks **cat nama\_daerah.txt | head -6** untuk menampilkan 6 baris string awal pada file.

****

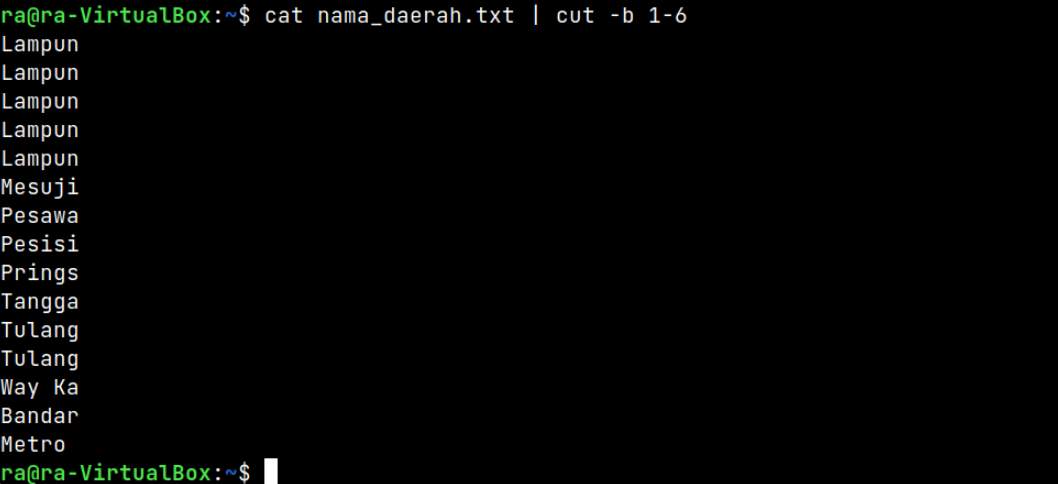
**2.3.6 Tail**

Untuk melakukan pipeline dan filter **Tail** pada file “nama\_daerah.txt” dapat menggunakan sintaks **cat nama\_daerah.txt | tail -6** untuk menampilkan 6 baris string akhir pada file.

****

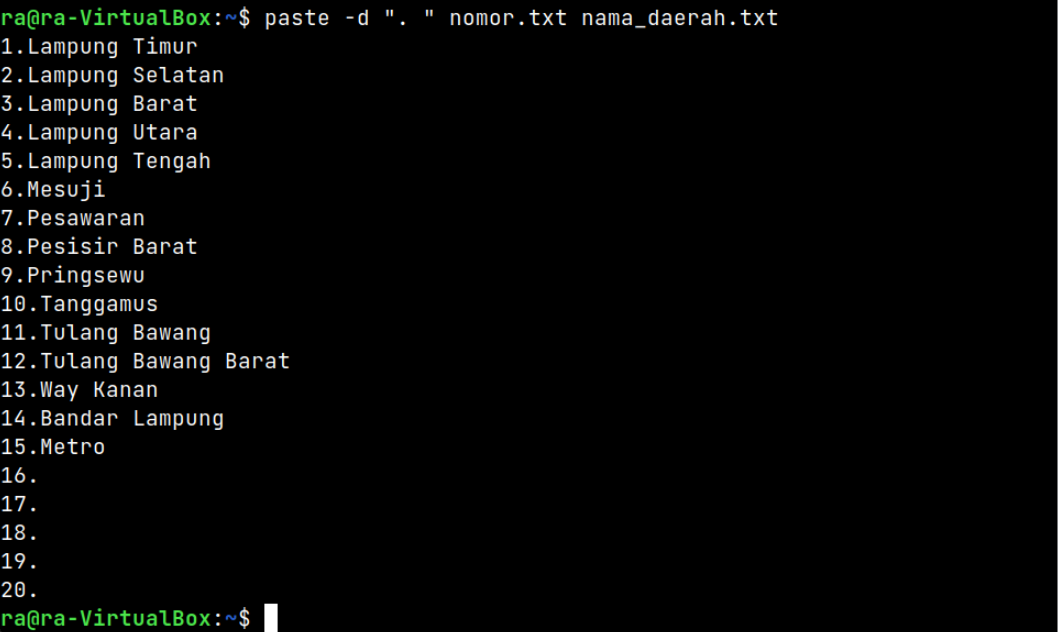
**2.3.7 Cut**

Untuk melakukan pipeline dan filter **Cut** pada file “nama\_daerah.txt” dapat menggunakan sintaks **cat nama\_daerah.txt | cut -b 1-6** untuk menampilkan 6 bit/string pada setiap baris dalam file.

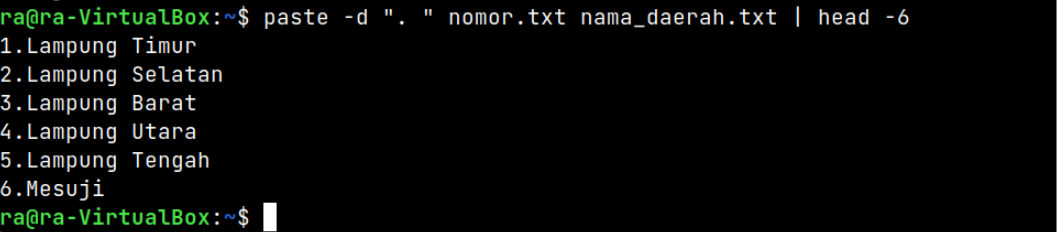
****

**2.3.8 Paste**

Untuk melakukan pipeline dan filter **Paste** file “nama\_daerah.txt” pada file “nomor.txt” menggunakan sintaks **paste -d “. ” nomor.txt nama\_daerah.txt** untuk menggabungkan isi dari file “nomor.txt” dengan “nama\_daerah.txt” dengan delimiter “. ” sehingga setelah file “nomor.txt” teroutputkan akan ada string “. ” sebelum file “nama\_daerah.txt” teroutputkan. Karena isi file dari “nama\_daerah.txt” hanya terdapat 15 baris, maka pada file “nomor.txt” baris 16-20 teroutputkan hanya dengan delimiter “. “.

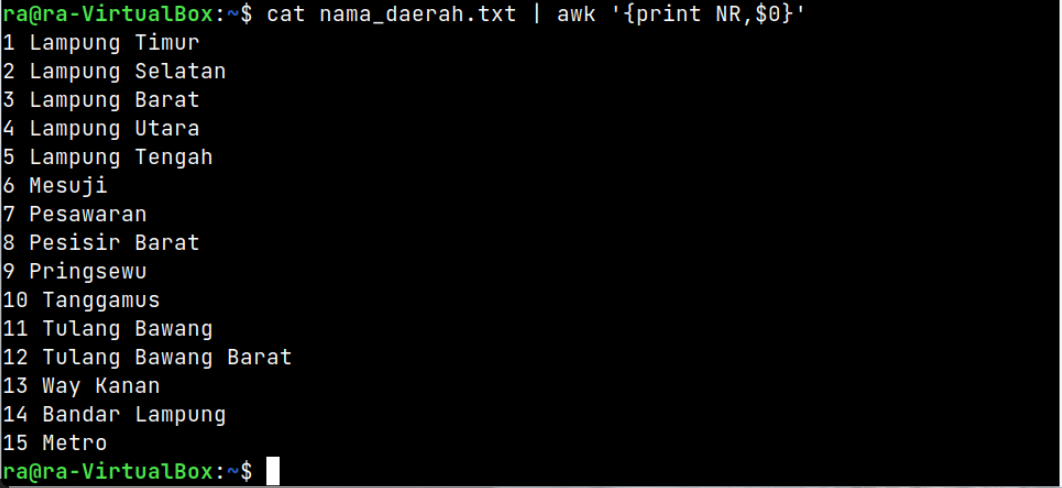
****

Jika ingin dikombinasikan dengan filter **Head** untuk membatasi outputnya hanya 6 baris awal saja dapat menggunakan sintaks **paste -d “. “ nomor.txt nama\_daerah.txt | head -6**, maka output file yang akan muncul tidak ada nomor dari “nomor.txt” yang melebihi baris pada file “nama\_daerah.txt”.

****

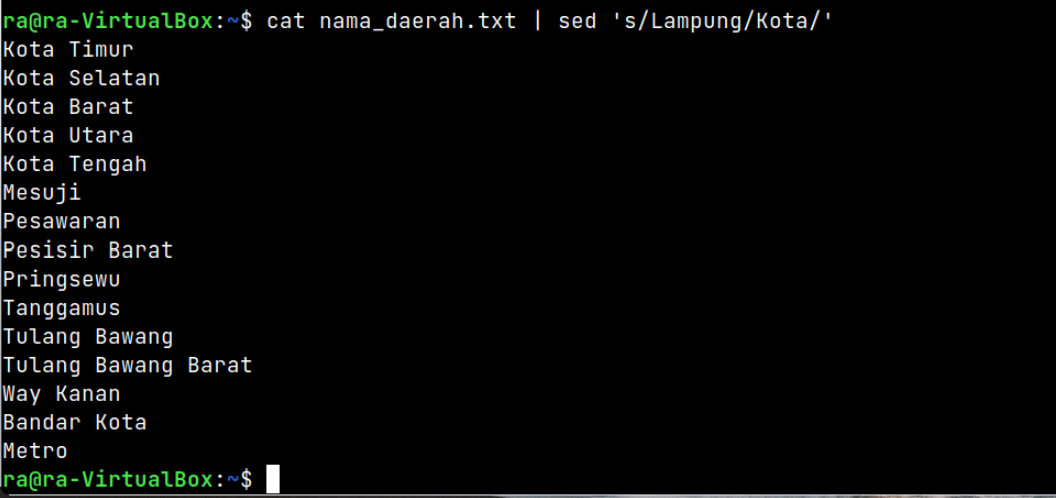
**2.3.9 Awk**

Untuk melakukan pipeline dan filter **Awk** pada file nama\_daerah.txt menggunakan sintaks **cat nama\_daerah.txt | awk ‘{print NR,$0}’** untuk menampilkan angka pada tiap baris file “nama\_daerah.txt”. Sintaks **NR** digunakan untuk menghitung dan mengoutputkan angka pada setiap baris secara berurutan, dan sintaks **$0** untuk mengoutputkan semua baris secara utuh.

****

**2.3.10 Sed**

Untuk melakukan pipeline dan filter **Sed** pada file “nama\_daerah.txt” menggunakan sintaks **cat nama\_daerahi.txt | sed ‘s/Lampung/Kota/’** untuk mengganti semua string yang mengandung string “Lampung” menjadi string “Kota”.

****

**BAB III  
KESIMPULAN**

Pada praktikum kali ini, dapat disimpulkan bahwa terdapat beberapa perintah yang digunakan untuk mengerjakan soal-soal di atas. Adapun perintahnya sebagai berikut :

* **nano** berfungsi untuk membuat file baru.
* **cat** berfungsi untuk menampilkan isi file yang ada pada direktori.
* **sudo su** berfungsi mengakses hak superuser
* **sudo apt install <nama-aplikasi>** berfungsi untuk menginstal aplikasi.
* **sudo apt-get update** berfungsi untuk memperbarui repository APT sebelum menginstal aplikasi
* **ps -e** berfungsi untuk melihat daftar semua proses yang sedang dijalankan di sistem Linux
* **ps -f** berfungsi untuk melihat format secara lengkap dari semua proses seperti ID user, nomor PID dan sebagainya.
* **ps –sort=-%cpu** berfungsi untuk mengurutkan proses sesuai dengan penggunaan cpu masing-masing proses.
* **ps -aux** memiliki fungsi yang hampir sama dengan **-f** hanya saja lebih detail dari **-f** karena merupakan gabungan dari beberapa parameter.
* **top -n 10**, dengan perintah ini, perintah teratas akan keluar secara otomatis setelah 10 kali pengulangan.
* **top -d <time>**, perintah ini berfungsi untuk memberi tahu waktu tunda antara pembaruan layar.
* **top -o %MEM** berfungsi untuk mengurutkan proses berdasarkan penggunaan memori (RAM). Biasanya lebih prioritaskan penggunaan memori yang lebih besar.