

# POKOK BAHASAN:

ﻫ Prinsip kerja redirection teks dan file

ﻫ Manipulasi standard input, output, dan error di shell

ﻫ Filter output atau file di shell menggunakan regular expression (regex)

ﻫ Menampilkan output file dengan paging atau berdasarkan pilihan

# OBJEKTIF:

* Mahasiswa mampu memahami prinsip kerja redirection
* Mahasiswa mampu mengimplementasikan perintah-perintah yang berkaitan dengan standard input, output, dan error
* Mahasiswa mampu mengimplementasikan perintah-perintah untuk melakukan filterisasi output atau file menggunakan regex
* Mahasiswa mampu menggunakan perintah- peritah untuk menampilkan output sesuai kebutuhan berbasis redirection dan regex

# TUGAS PRAKTIKUM:

1. Mengerjakan setiap langkah yang ada di modul praktikum dengan bukti print screen dari output yang didapat dan harus melalui shell dengan nama Anda.
2. Menjawab seluruh soal praktikum
3. Mengumpulkan hasil praktikum di learning pada link yang telah disediakan dengan format file pdf, berikan nama file dengan nama Anda

# TIDAK ADA PERPANJANGAN WAKTU PENGUMPULAN



**Jangan lupa copy-kan file vdi hasil instalasi Linux Anda ke Flashdisk Anda agar Anda dapat menggunakan kembali hasil instalasi tersebut di komputer manapun cukup dengan menginstal kembali dari file vdi ini!!**

**PENJELASAN:**

Secara normal ketika kita menuliskan perintah di shell kemudian dieksekusi maka akan tampil output di layar, (output ini disebut juga dengan istilah *channel* ). Output yang tampil di layar disebut dengan **standard output** yang ditulisan dengan keterangan **stdout**, dilambangkan dengan simbol **>**, dan dengan nomor **deskriptor file** untuk channel ini adalah **1**.

**Standard error** terjadi ketika ada sebuah perintah(*command* ) yang salah pada saat dieksekusi, yang dilambangkan dengan istilah/tulisan **stderr**, dilambangkan dengan simbol **2>**, dan dengan nomor **deskriptor file** untuk channel ini adalah **2** dan pesan ini juga akan dikirim ke layar output secara default.

**Standard input** biasaya pemberian input/perintah di shell dilakukan oleh user melalui keyboard, namun dengan *redirecting* (pengarahan) standard input maka file juga dapat digunakan sebagai standard input. **Standard input** yang ditulis dengan keterangan **stdin,** dilambangkan dengan **<** dan dengan nomor **deskriptor file** untuk channel ini adalah **0**.

# LANGKAH 1:

Menggunakan simbol redirection **stdout** (pengarahan) **>** yang tujuannya untuk mengarahkan output dari output normal yang seharusnya tampil di layar menjadi diarahkan ke sebuah file.

Sekarang ketikkan perintah-perintah berikut ini di shell:

1. **cd** : ke lokasi **home directory**
2. **ls** : melihat isi **home directory**
3. **echo “Hello World!”** : menampilkan output teks **Hello World!** di Shell
4. **echo “Hello World!” > pesanku** : mengarahkan output teks **Hello World!** ke file

dengan nama pesanku

1. **ls** : lihat kembali isi **home directory** dan di sana sudah ada file baru dengan nama **pesanku**
2. **cat pesanku** : menampilkan isi file pesanku Screenshot hasilnya!

# LANGKAH 2:

Sekarang berikan kembali isi untuk file pesanku, ketikkan perintah berikut ini di shell:

1. **cat pesanku**
2. **echo apa kabar > pesanku**
3. **cat pesanku**

Screenshot hasilnya!

Ketika kita berikan perintah **cat**, ternyata isi file pesanku menjadi **apa kabar**, lalu kemana kata **Hello World!** yang sebelumnya? Penggunaan simbol **>** ini mengakibatkan isi file di overwrite/ditimpa sehingga isi file yang lama akan diganti dengan isi yang baru, istilah ini disebut dengan “*clobbering*” (musnah) file.

# LANGKAH 3:

Cara agar isi file tidak ditimpa dengan isi yang baru, kita dapat menggunakan simbol **>>** untuk menambahkan isi di akhir file.

Sekrang ketikkan perintah berikut ini di shell:

1. **echo semoga kalian semua dalam keadaan baik >> pesanku**
2. **cat pesanku**

Screenshot hasilnya!

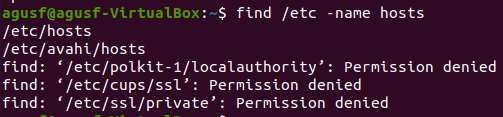
Apakah teks **semoga kalian semua dalam keadaan baik** ada di baris bawah kata **apa kabar**?

# LANGKAH 4:

Perintah **find** adalah contoh perintah yang bagus untuk menunjukan kerja dari **stderr**. **Find** adalah perintah yang digunakan untuk mencari filesystem yang didasarkan pada kriteria pencarian yang diberikan seperti nama file.

Sekarang ketikkan perintah berikut ini di shell:

**find /etc –name hosts** maka output akan tampil seperti yang terlihat pada gambar 8.1



Gambar 8.1 Perintah Find dengan Hasil Error

Dari gambar 8.1 di atas ada pesan **error** yang menggambarkan bahwa user tersebut tidak memiliki hak akses terhadap file/direktori tertentu. Tipe pesan error ini dikirim ke **stderr** bukan ke **stdout**.

Pesan error yang terjadi di langkah 4 dapat dialihkan (*redirect*) ke sebuah file tertentu. Sekarang ketikkan perintah berikut ini di shell:

1. **find /etc –name hosts 2> pesan\_salah.txt**
2. **ls**
3. **cat pesan\_salah.txt**

Screenshot hasilnya!

Penjelasan:

**find** : Adalah perintah untuk melakukan pencarian file/direktori

**/etc** : Direktori yang berisi file yang akan dicari

**-name** : Opsi untuk menspesifikasikan nama file yang akan dicari

**hosts** : Nama file yang dicari

**2>** : Simbol *redirection stderr*, pesan error dialihkan ke file **pesan\_salah.txt pesan\_salah.txt** : Nama file yang diciptakan untuk menampung teks error

# LANGKAH 6:

Kita juga dapat memisahkan secara bersamaan antara **stdout** dan **stderr** dalam satu perintah, jika Anda lihat gambar 8.1 di sana terlihat bahwa ada output **/etc/hosts** dan **/etc/avahi/hosts** yang kemudian baru diikuti pesan errornya yang berupa **Permission denied**.

Tahukan Anda bahwa sebenarnya keluaran dari perintah tersebut terdiri dari 2 keluaran, yaitu yang pertama sebagai **stdout** untuk **/etc/hosts** dan **/etc/avahi/hosts** lalu yang kedua sebagai **stderr** untuk informasi errornya.

Penggabungan perintah untuk mengarahkan hasil yang didapat ke masing-masing keluarannya dapat dilakukan dengan mengikuti langkah-langkah berikut ini:

1. **find /etc -name hosts > file.stdout 2> file.stderr**
2. **ls**
3. **cat file.stdout**
4. **cat file.stderr**

Screenshot hasilnya!

Normalnya **stdin** datang melalui inputan yang diberikan oleh user melalui keyboard, tapi kadang

**stdin** juga dapat datang dari sebuah file.

Sekarang ketikkan perintah berikut ini di shell:

1. **tr a-z A-Z** : **tr** dari kata translate, mengubah inputan dari huruf kecil menjadi huruf besar
2. **rekayasa keamanan siber** : teks yang dijadikan inputan
3. tekan enter
4. tekan Ctrl+D Screenshot hasilnya!

Menjalankan aktivitas **stdin** dengan menggunakan simbol **<** dapat juga dilakukan dari sebuah file. Gunakan file **pesanku** seperti yang telah Anda buat lalu jalankan perintah berikut di shell: **tr a-z A-Z < pesanku** dan screenshot hasilnya!

# LANGKAH 8:

Bentuk popular lainnya dari *redirection* ini yaitu *pipe line* yang disimbolkan dengan vertikal bar **|**, ketika sebuah perintah dipisahkan dengan *pipe line* itu artinya output dari perintah pertama akan dikirim ke perintah berikutnya sebagai inputan.

Sekarang ketikkan perintah berikut di shell:

1. **ls /etc**
2. **ls /etc | more**

Screenshot hasilnya!

Apa perbedaan dari 2 perintah di atas, jelaskan!

# LANGKAH 9:

**PCRE** : penggunaannya memerlukan karakter khusus dan lebih banyak dari yang dimiliki BRE dan ERE

Perintah **find** adalah perintah yang sangat fleksibel dengan sejumlah opsi yang memungkin pengguna menemukan file berdasarkan beragam kriteria seperti nama file, tanggal, ukuran, jenis, dan hak akses.

Format perintah find adalah

**find <lokasi pencarian> -opsi/kriteria <argumen kriteria>**

Jalankan perintah berikut:

1. **find ~ -name “\*bash\*”** : mencari file berdasarkan nama
2. **find ~/ -mmin -120** : mencari file berdasarkan waktu di bawah 120 menit
3. **find ~ -size -2M** : mencari file berdasarkan ukuran di bawah 2M
4. **find ~ -type d** : mencari file berdasarkan tipe file direktori Screenshot hasilnya!

# LANGKAH 10:

**Tabel 8.1 Variasi Perintah grep**

**grep (biasa) :** Perintah **grep** umum dan tanpa karakter khusus ketika melakukan pencocokan stringnya

**egrep**

: Perintah **egrep** atau dapat menggunakan perintah **grep** dengan opsi **–E** dan membutuhkan

karakter khusus ketika melakukan pencocokan stringnya seperti: **?**, **+**, **|**

**fgrep** : Perintah **fgrep** atau dapat menggunakan perintah **grep** dengan opsi **–F** adalah perintah untuk melakukan pencocokan string yang mengandung karakter-karakter khusus seperti **$**, **^**, **.**, atau yang yang lainnya

**rgrep**

: Perintah **rgrep** atau dapat menggunakan perintah **grep** dengan opsi **–r** untuk melakukan

pencocokan string secara rekursif mulai dari awal direktori yang ditentukan

Variasi kategori bentuk perintah **grep** diantaranya: **grep**, **basic regular expression (BRE), extended regular expression (ERE),** dan **perl regular expression (PCRE)** untuk PCRE sendiri tidak masuk dalam pembahasan. Penjelasan mengenai perbedaan dari masing-masing kategori seperti terlihat pada tabel 6.2 variasi kategori grep.

**Tabel 8.2 Variasi Kategori grep**

**grep (biasa) :** penggunaannya tanpa memerlukan karakter khusus

**BRE** : penggunaannya memerlukan karakter khusus seperti: **.**, **[]**, **^**, **\***, **$**

**ERE** : penggunaannya memerlukan karakter khusus seperti: **?**, **+**, **|, { }**

Penggunaan perintah **grep** dalam bentuk yang sederhana untuk mencari rangkaian karakter tertentu. Misal kita ingin mencari kata **mail** di file **/etc/passwd**, maka perintahnya adalah: **grep mail /etc/passwd**, maka akan tampil output seperti yang terlihat pada gambar 8.2



Gambar 8.2

Penjelasan:

**grep** : Adalah perintah utama untuk melakukan pencarian string dalam file **mail** : String atau rangkaian karakter yang dicari

**/etc/passwd** : Lokasi pencarian string dari file passwd **LANGKAH 11:**

lalu bagaimana caranya agar yang dicocokkan misalnya hanya kata yang di awal baris saja? Perintah

grep biasa tidak memiliki kemampuan untuk melakukan itu, maka dari itu kita butuh karakter khusus yang dapat menyelesaikan permasalahan ini yaitu dengan memberikan tanda **^ (karet)** di awal pola/kata yang ingin dicari. Sekarang ketikkan perintah ini di shell: **grep ^mail**

**/etc/passwd**, maka akan tampil output seperti yang terlihat pada gambar 8.3



Gambar 8.3

Pemberian karakter khusus dengan simbol ^ ini menandakan bahwa kita sedang menggunakan grep dengan kategori BRE.

Penjelasan:

**grep** : Adalah perintah utama untuk melakukan pencarian string dalam file

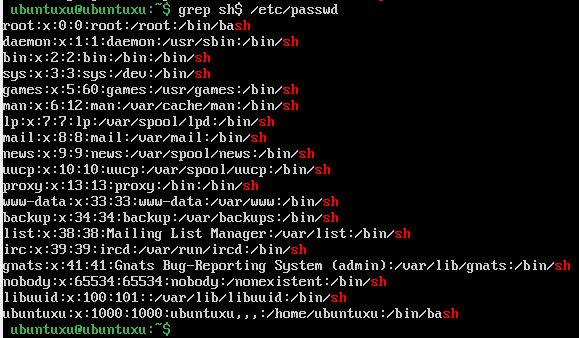
**^** : Simbol karet yang merupakan bagian dari simbol BRE yang perannya untuk mencocokan string di awal baris

**mail** : String atau rangkaian karakter yang dicari

**/etc/passwd** : Lokasi pencarian string dari file passwd

# LANGKAH 12:

Cara untuk menampilkan agar yang dicocokkan hanya kata yang di akhir baris saja, dapat dilakukan dengan memberikan tanda **$ (dollar)** di akhir pola/kata yang ingin dicari. Sekarang ketikkan perintah ini di shell: **grep sh$ /etc/passwd**, maka akan tampil output seperti yang terlihat pada gambar 8.4



Gambar 8.4

Penjelasan:

**grep** : Adalah perintah utama untuk melakukan pencarian string dalam file

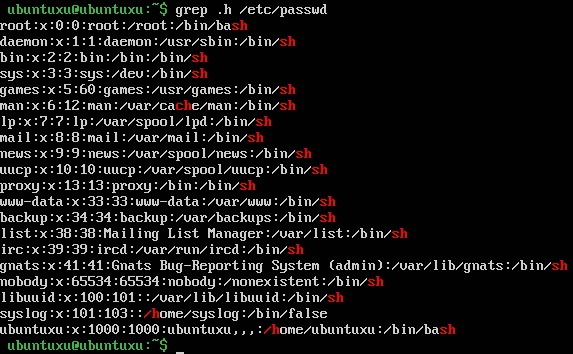
**$** : Simbol dollar yang merupakan bagian dari simbol BRE yang perannya untuk mencocokan string di akhir baris

**sh** : String atau rangkaian karakter yang dicari

**/etc/passwd** : Lokasi pencarian string dari file passwd **LANGKAH 13:**

Cara untuk mencari per karakter yang diikuti oleh pola/kata yang ingin dicari dapat dilakukan

dengan memberikan tanda **. (titik)**. Sekarang ketikkan perintah ini di shell: **grep .h /etc/passwd**, maka akan tampil output seperti yang terlihat pada gambar 8.5



Gambar 8.5

Dari gambar 8.5 terlihat ada satu karakter yang dicocokkan di depan karakter h tidak perduli karakternya apa, selama posisinya berada tepat di depan karakter h maka akan ditampilkan, dari gambar tersebut terlihat ada karakter s,c,dan /. **Satu titik** menandakan bahwa yang ingin dicari **hanya satu karakter** saja, jika **dua titik** berarti **dua karakter**, **tiga titik** berarti **tiga karakter**, dan seterusnya, banyak jumlah titik menandakan banyaknya jumlah karakter yang ingin dicari.

Penjelasan:

**grep** : Adalah perintah utama untuk melakukan pencarian string dalam file

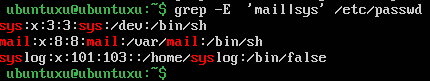
**.** : Simbol titik yang merupakan bagian dari simbol BRE yang perannya untuk mencocokan karakter dari pola yang telah ditentukan

**sh** : String atau rangkaian karakter yang dicari

**/etc/passwd** : Lokasi pencarian string dari file passwd

# LANGKAH 14:

Cara untuk mencari beberapa kata sekaligus dari sebuah file dapat dilakukan dengan memberikan tanda **| (pipe line)**. Sayang nya perintah grep standar tidak dapat melakukan hal tersebut, oleh karenanya kita butuh teknik lain untuk menyelesaikan permasalah ini yaitu menggunakan variasi perintah **grep** dengan opsi **–E** atau yang dikenal dengan istilah **egrep**. Sekarang ketikkan perintah ini di shell: **grep –E ‘mail|sys’ /etc/passwd**, maka akan tampil output seperti yang terlihat pada gambar 8.6



Gambar 8.6

Penjelasan:

**grep** : Adalah perintah utama untuk melakukan pencarian string dalam file

**-E** : Opsi dari perintah grep yang menyatakan extended, artinya butuh karakter khusus untuk dapat menampilkan kata yang dicari

**‘ ’** : Simbol petik tunggal diberikan untuk mencegah shell menerjemahkan menurut pengertiannya

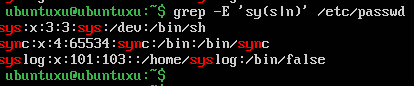
**mail atau sys** : Kata yang ingin dicari/dicocokkan

**|** : Simbol pipe line yang merupakan bagian dari simbol ERE yang perannya untuk memberikan alternatif dari kata yang ingin dicari berdasarkan pola yang telah ditentukan

**/etc/passwd** : Lokasi pencarian string dari file passwd

# LANGKAH 15:

Cara untuk mencari dua string sekaligus dapat dilakukan masih dengan memanfaatkan karakter **| (pipe line)**, hanya saja bagian karakter yang jadi pembeda antara string yang satu dengan yang lainnya harus diletakkan di dalam kurung. Sekarang ketikkan perintah ini di shell: **grep –E ‘sy (s|n)’ /etc/passwd**, maka akan tampil output seperti yang terlihat pada gambar 8.7



Gambar 8.7

Penjelasan:

**grep** : Adalah perintah utama untuk melakukan pencarian string dalam file

**-E** : Opsi dari perintah grep yang menyatakan extended, artinya butuh karakter khusus untuk dapat menampilkan kata yang dicari

**‘ ’** : Simbol petik tunggal diberikan untuk mencegah shell menerjemahkan menurut pengertiannya

**sy(s|n)** : Kata yang ingin dicari/dicocokkan, dan akan menampilkan string **sys** atau **syn**

**|** : Simbol pipe line yang merupakan bagian dari simbol ERE yang perannya untuk memberikan alternatif dari kata yang ingin dicari berdasarkan pola yang telah ditentukan

**( )** : Tanda kurung digunakan untuk membatasi ruang lingkup pencarian karakter, jika tanpa tanda kurung maka yang dicocokkan adalah kata sys atau n, sehingga

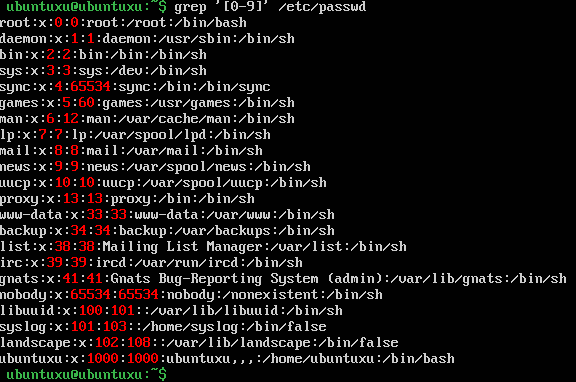
tidak sesuai tujuan

**/etc/passwd** : Lokasi pencarian string dari file passwd **LANGKAH 16:**

Cara untuk mencari karakter numerik (angka) dapat dilakukan menuliskan angka-angka yang ingin

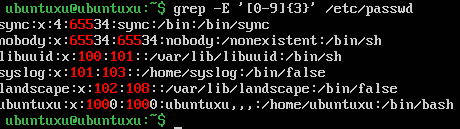
dicari atau berdasarkan range angka di dalam kurung siku. Sekarang ketikkan perintah ini di shell:

**grep ‘[0-9]’ /etc/passwd**, maka akan tampil output seperti yang terlihat pada gambar 8.8



Gambar 8.8

Dari gambar di atas terlihat bahwa semua angka yang ada file passwd akan dicocokkan semuanya berapapun jumlah angkanya, lalu bagaimana caranya agar yang ditampilkan jumlah digitnya dibatasi, misalnya dua digit saja, atau tiga digit, atau empat digit, dan seterusnya? Untuk dapat melakukan hal tersebut kita harus menggunakan kategori ERE dengan cara memberikan batasan berapa banyak pengulangan digit yang ingin dicocokan. Misal kita ingin mencocokan pengulangan tiga digit saja maka kita menggunakan tanda kurung kurawal dengan memberikan angka di dalamnya yang menyatakan banyaknya jumlah pengulangan digit yang akan dicocokkan. Kurung kurawal ini merupakan karakter khusus yang masuk dalam kategori ERE. Sekarang ketikkan perintah ini di shell: **grep -E ‘[0-9]{3}’ /etc/passwd**, maka akan tampil output seperti yang terlihat pada gambar 8.9



Gambar 8.9

Dari gambar 8.9 di atas, sekarang terlihat bahwa digit yang dicocokkan hanya tiga digit pertama saja dari urutan angka yang ada di file passwd.

# LANGKAH 17:

Xargs adalah salah satu perintah di Linux yang fungsinya membantu kumpulan perintah pipeline yang kompleks menjadi lebih efisien dalam pengeksekusiannya. Xargs sendiri singkatan dari extended arguments, artinya di sini dapat menggunakan banyak argumen sekaligus dalam pemberian perintahnya dan mencegah eksekusi perintah untuk setiap argumen yang diberikan dengan kata lain eksekusinya dibuat sekaligus.

Ikuti langkah-langkah ini: CARA I

1. **mkdir Linux**, tekan enter (membuat direktori baru dengan nama Linux)
2. **cd Linux**, tekan enter (masuk ke direktori Linux)
3. **mkdir Satu**, tekan enter (membuat direktori baru dengan nama Satu di dalam direktori Linux)
4. **mkdir Dua**, tekan enter (membuat direktori baru dengan nama Dua di dalam direktori Linux)
5. **mkdir Tiga**, tekan enter (membuat direktori baru dengan nama Tiga di dalam direktori Linux)
6. **ls**, tekan enter, maka akan terlihat ada direktori yang sudah dibuat tadi

Ikuti langkah-langkah ini: CARA II

pastikan direktori aktif Anda masih berada di direktori Linux!

1. **echo Empat Lima Enam | xargs mkdir**, tekan enter
2. **ls**, tekan enter

Lihat apa yang tampil di dalam direktori Linux?. *Amazing*, sekarang sudah ada direktori baru dengan nama Empat, Lima, dan Enam, itu artinya dengan perintah xargs kita dapat melakukan pemberian banyak argumen sekaligus dan dapat langsung dieksekusi dengan hasil yang lebih efisien!. Dari dua cara yang digambarkan di atas menunjukkan bahwa dengan perintah xargs terbukti lebih efisien.

# PERTANYAAN:

1. Apa simbol untuk redirection input/stdin dan kode file deskriptor nya?
2. Apa simbol untuk redirection output/stdout dan kode file deskriptor nya?
3. Apa simbol untuk redirection error/stderr dan kode file deskriptor nya?
4. Apakah perintah pada langkah 6 jika perintahnya menjadi **find /etc –name hosts 2>file.stderr >file.stdout** dapat dieksekusi dan menghasilkan isi yang sama? Buktikan!
5. Lakukan seperti pada langkah 8, tulisan kalimatnya adalah “MISTERI YANG KAU SIMPAN MEMBUAT AKU SEMAKIN TERPESONA” hanya saja hasil translate dari stdin harus dirubah menjadi huruf kecil dan kemudian dikirim ke stdout dengan nama file LinuxKu.
6. Tampilkan isi dari file LinuxKu menggunakan perintah **cat**!
7. Seandainya Anda ganti perintah more dengan less di langkah 10, apakah ada perbedaan hasil?
8. Lakukan seperti di langkah 11, hanya saja tampilan outputnya dua kolom pertama dari file

/etc/passwd tersebut! Perintah apa yang harus Anda berikan!

1. Apa perintah yang harus Anda berikan untuk mencari file dengan nama hosts! Tampilan output di layar nya tidak boleh mengandung tampilan error.
2. Apa perintah yang harus Anda berikan untuk mencari file berdasarkan perubahan dalam hitungan hari?
3. Apakah proses pencarian file di satu lokasi direktori akan diteruskan ke subdirektori di bawahnya? Atau perlu ada opsi tambahan untuk mencari file yang ada di subdirektorinya?
4. Cari ukuran file yang berada di bawah home direktori Anda yang ukurannya dalam besaran byte dan kilo byte!
5. Tampilkan semua file regular (BUKAN DIREKTORI) yang berada di bawah home direktori Anda namun sekaligus dapat menghitung berapa banyak file yang di dapat! Seperti terlihat pada gambar soal 13



Gambar soal 13

1. Apakah simbol ^ dan simbol dollar dapat digabungkan disatu kata yang sama pada perintah grep?
2. Apa perintah untuk mencari dua karakter di depan karakter **o (bukan nol)** dari file

/etc/passwd!

1. Jika Anda perhatikan tampilan gambar 6.22 di langkah 22, di sana terlihat teks **syslog** yang bagian **sys** nya ditandai sebagai string yang dicocokkan. Sekarang apa perintah untuk menampilkan string yang memang benar-benar dari satu kata yang utuh?, string **sys** dari **kata syslog tidak boleh tampil** dari file /etc/passwd ketika dieksekusi! **Clue**: manfaatkan halaman manual dari perintah grep untuk menyelesaikan kasus ini.
2. Ganti salah satu perintah **grep –E** dari langkah yang ada di atas dengan perintah **egrep**, apakah menghasilkan output yang sama?
3. Lakukan penghapusan direktori yang berada di dalam direktori Linux secara sekaligus menggunakan perintah **xargs**!

# KESIMPULAN:

Berikan kesimpulan Anda dari praktikum ini!

*tuliskan ada berapa perintah yang sudah Anda kuasai ditambah dengan yang lalu*

# SARAN:

Berikan saran jika ada!