

LAPORAN PRAKTIKUM
SISTEM OPERASI
SHELL PROGRAMMING
PERTEMUAN KE-7



Disusun Oleh :
NAUFAL FADHIL IHSAN FIKRI ASH-SHIDIQI
2250081109

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS SAINS DAN INFORMATIKA
UNIVERSITAS JENDERAL ACHMAD YANI
2024

BAB I

HASIL PRAKTIKUM

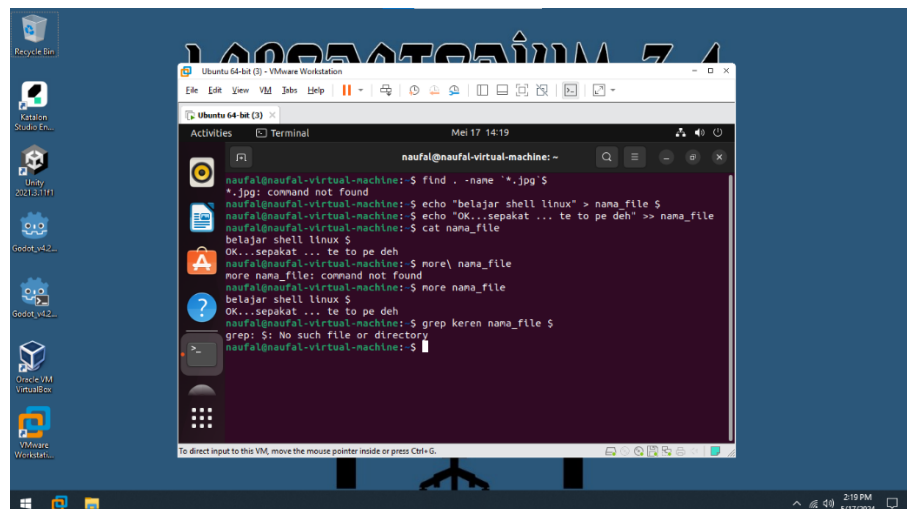
1.1 Terminal Direktori

Percobaan 1 : Shell Interaktif

a. Source Code

```
$ find . -name '*.jpg' $  
$ echo "belajar shell linux" > nama_file $  
echo "keren abis" >> nama_file  
$ echo "OK...sepakat .... te o pe deh" >> nama_file  
$ cat nama_file  
$ more nama_file  
$ grep keren nama_file $
```

b. Tampilan Program



c. Analisa

Perintah `find . -name '*.jpg'` digunakan untuk mencari file dengan ekstensi `.jpg` di dalam direktori saat ini dan semua subdirektori yang terkait, menampilkan daftar file JPG yang ditemukan dalam struktur direktori yang diperiksa. Perintah `echo "belajar shell linux" >`

nama_file` membuat file baru bernama `nama_file` dan menulis teks "belajar shell linux" ke dalamnya, dengan tanda `>` digunakan untuk mengarahkan output teks ke dalam file, menggantikan isi file jika sudah ada atau membuat file baru jika belum ada. Perintah `echo "keren abis" >> nama_file` menambahkan teks "keren abis" ke dalam `nama_file`, dengan tanda `>>` digunakan untuk menambahkan teks ke akhir file tanpa menghapus isi yang sudah ada, dan perintah `echo "OK...sepakat te o pe deh" >> nama_file` juga menambahkan teks ke dalam `nama_file`, menambahkan teks baru ke baris terakhir. Perintah `cat nama_file` menampilkan isi dari file `nama_file`, memungkinkan pengguna untuk melihat seluruh konten file sekaligus, sementara perintah `more nama_file` menampilkan isi file `nama_file` dalam format yang lebih interaktif, memungkinkan pengguna untuk membaca konten file satu layar pada satu waktu. Perintah `grep keren nama_file` digunakan untuk mencari kata "keren" dalam `nama_file` dan menampilkan baris-baris di mana kata tersebut ditemukan, dengan `grep` sebagai alat pencarian yang berguna untuk menemukan pola atau kata dalam teks.

Percobaan 2 : Script Shell

a. Source Code

```
$ nano Latihan1.sh
#!/bin/bash
echo "selamat datang Sandro Anugrah Tambunan"
echo "di shell programming"
chmod +x Latihan1.sh
./Latihan1.sh

$ nano Latihan2.sh
#!/bin/bash
echo "Shell yang digunakan adalah $SHELL"
echo "saat ini jam `date +%T`"
echo "tanggal `date +%D`"
chmod +x Latihan2.sh
./Latihan2.sh

$ nano Latihan3.sh
#!/bin/bash
clear
echo "komputer anda telah menyala selama `uptime`"
echo "jumlah user yang login sebanyak `who | wc -l` user"
echo "anda login dengan user $LOGNAME"
echo "di shell $SHELL"
chmod +x Latihan3.sh
./Latihan3.sh
```

```
$ nano Latihan4.sh

#!/bin/bash

TITEL="Membuat Fungsi Sistem Informasi $HOSTNAME"

SAAT_INI=$(date +"%d %T %Z") # tambahkan spasi
setelah 'date' dan sebelum '%'

UPD="Sistem ini di update oleh $USER pada tanggal
$SAAT_INI"

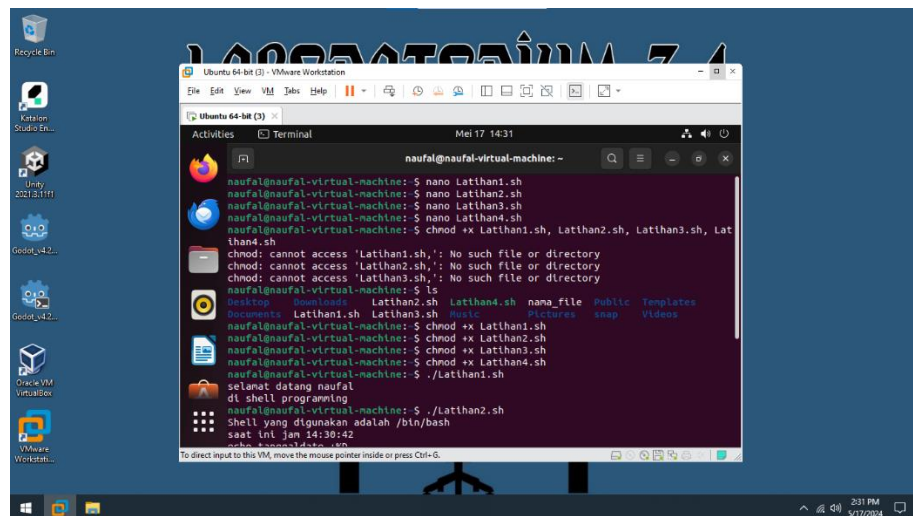
function info_uptime(){
    echo "<h2>informasi uptime</h2>"
    echo "<pre>"
    uptime
    echo "</pre>"
}

cat <<- EOF
<HTML>
<HEAD>
<TITLE>$TITEL</TITLE>
</HEAD>
<BODY>
<H1>$TITEL</H1>
<P>$UPD</P>
$(info_uptime)
</BODY>
</HTML>
EOF

chmod +x Latihan4.sh

./Latihan4.sh
```

b. Tampilan Program



```
naufal@naufal-virtual-machine: ~  
naufal@naufal-virtual-machine: $ nano Latihan1.sh  
naufal@naufal-virtual-machine: $ nano Latihan2.sh  
naufal@naufal-virtual-machine: $ nano Latihan3.sh  
naufal@naufal-virtual-machine: $ nano Latihan4.sh  
naufal@naufal-virtual-machine: $ chmod +x Latihan1.sh, Latihan2.sh, Latihan3.sh, Latihan4.sh  
naufal@naufal-virtual-machine: $ ls  
Desktop Downloads Latihan1.sh Latihan2.sh Latihan3.sh Latihan4.sh naufal_file Public Templates  
Documents Latihan1.sh Latihan3.sh Music Pictures snap Videos  
naufal@naufal-virtual-machine: $ chmod +x Latihan1.sh  
naufal@naufal-virtual-machine: $ chmod +x Latihan2.sh  
naufal@naufal-virtual-machine: $ chmod +x Latihan3.sh  
naufal@naufal-virtual-machine: $ chmod +x Latihan4.sh  
naufal@naufal-virtual-machine: $ ./Latihan1.sh  
selamat datang naufal  
naufal@naufal-virtual-machine: $ ./Latihan2.sh  
Shell yang digunakan adalah /bin/bash  
saat ini jam 14:38:42  
naufal@naufal-virtual-machine: $
```

c. Analisa

Perintah `nano Latihan1.sh` digunakan untuk membuat atau mengedit skrip shell `Latihan1.sh`, yang menampilkan pesan "selamat datang Sandro Anugrah Tambunan di shell programming" ke layar. Setelah membuat skrip, perintah `chmod +x Latihan1.sh` diberikan untuk memberikan izin eksekusi pada skrip tersebut, kemudian skrip dijalankan dengan `./Latihan1.sh`. Perintah `nano Latihan2.sh` digunakan untuk membuat atau mengedit skrip shell `Latihan2.sh`, yang menampilkan informasi tentang shell yang digunakan, jam saat ini, dan tanggal saat ini. Setelah membuat skrip, perintah `chmod +x Latihan2.sh` diberikan untuk memberikan izin eksekusi pada skrip tersebut, kemudian skrip dijalankan dengan `./Latihan2.sh`. Perintah `nano Latihan3.sh` digunakan untuk membuat atau mengedit skrip shell `Latihan3.sh`, yang menampilkan informasi tentang waktu komputer telah menyala, jumlah pengguna yang login, nama pengguna yang sedang login, dan jenis shell yang digunakan. Setelah membuat skrip, perintah `chmod +x Latihan3.sh` diberikan untuk memberikan izin eksekusi pada skrip tersebut, kemudian skrip dijalankan dengan `./Latihan3.sh`. Perintah `nano Latihan4.sh` digunakan untuk membuat atau mengedit skrip shell `Latihan4.sh`, yang membuat fungsi `info_uptime()` untuk menampilkan informasi tentang uptime sistem dan

mencetak informasi tentang judul, update, dan hasil pemanggilan fungsi ``info_uptime()`` ke dalam format HTML. Setelah membuat skrip, perintah ``chmod +x Latihan4.sh`` diberikan untuk memberikan izin eksekusi pada skrip tersebut, kemudian skrip dijalankan dengan ``./Latihan4.sh``.

Tugas

Analisis sudah ada di hasil praktikum

Tugas Tambahan

1. Jelaskan perbedaan utama antara beberapa jenis shell yang populer di Linux seperti Bash, Zsh, dan Fish. Apa kelebihan dan kekurangan masing-masing shell tersebut? Berikan contoh situasi di mana kamu akan memilih satu jenis shell dibandingkan yang lain. 2. Bagaimana cara mengganti shell default pengguna di Linux? Jelaskan langkah-langkahnya secara detail, dan sebutkan perintah yang digunakan. Apa risiko dan pertimbangan yang harus diperhatikan ketika mengganti shell default?

2. Jelaskan perbedaan antara mode interaktif dan non-interaktif dalam penggunaan shell. Bagaimana mode ini mempengaruhi cara eksekusi perintah dan script? Berikan contoh kasus di mana perbedaan ini menjadi penting.

4. Buatlah sebuah script shell sederhana yang melakukan backup file tertentu ke lokasi yang ditentukan pengguna. Script harus menerima input nama file dan lokasi backup, serta menampilkan pesan sukses atau gagal setelah proses backup selesai.

4. Diskusikan bagaimana script shell dapat digunakan untuk otomatisasi tugas-tugas administrasi sistem di Linux. Berikan contoh nyata dari sebuah tugas yang dapat diotomatisasi menggunakan script shell dan jelaskan langkah-langkah dalam script tersebut.

1. Perbedaan antara Bash, Zsh, dan Fish

Bash (Bourne Again Shell)

- Kelebihan:

- ****Kompatibilitas****: Bash adalah shell default di banyak distribusi Linux, sehingga banyak skrip dan tool dirancang untuk bekerja dengan Bash.
- **Dokumentasi**: Bash memiliki dokumentasi yang sangat luas dan komunitas besar yang dapat memberikan bantuan.
- **Stabilitas**: Karena merupakan shell standar, Bash sangat stabil dan terpercaya untuk berbagai tugas.

- Kekurangan:

- **Fitur terbatas**: Tidak se-modern beberapa shell lainnya dalam hal fitur interaktif dan auto-suggestions.

- **Contoh Penggunaan**: Ketika menulis skrip yang perlu berjalan di berbagai sistem yang berbeda, karena hampir semua sistem Linux memiliki Bash yang sudah terpasang.

Zsh (Z Shell)

- Kelebihan:

- **Fitur Interaktif**: Zsh memiliki fitur seperti auto-suggestions, auto-completion yang lebih canggih, dan kemampuan untuk mengkustomisasi prompt dengan mudah.
- **Oh-My-Zsh**: Framework ini membuat Zsh sangat mudah untuk dikustomisasi dan ditingkatkan dengan berbagai plugin.

- Kekurangan:

- **Kompatibilitas**: Tidak sekompatibel Bash, jadi beberapa skrip mungkin memerlukan penyesuaian.
- **Learning Curve**: Memiliki lebih banyak fitur yang mungkin memerlukan waktu untuk dipelajari.

- Contoh Penggunaan: Saat membutuhkan shell dengan fitur interaktif yang kaya untuk penggunaan sehari-hari atau ketika ingin memanfaatkan plugin dari Oh-My-Zsh.

Fish (Friendly Interactive Shell)

- Kelebihan:

- User-Friendly: Sangat mudah digunakan, dengan sintaks yang lebih bersih dan auto-suggestions yang sangat kuat.
- Out-of-the-box Features: Fish menyediakan banyak fitur yang bekerja langsung tanpa konfigurasi tambahan.

- Kekurangan:

- Kompatibilitas Skrip: Fish tidak sepenuhnya kompatibel dengan skrip Bash, jadi mungkin perlu perubahan besar untuk menjalankan skrip yang ditulis untuk Bash.
- Kustomisasi: Tidak sefleksibel Zsh dalam hal plugin dan kustomisasi.

- Contoh Penggunaan: Ketika menginginkan pengalaman shell yang ramah pengguna dan siap digunakan dengan fitur-fitur modern.

2. Mengganti Shell Default Pengguna di Linux

Langkah-langkah:

1. ****Lihat Shell yang Tersedia****:

```
``sh
```

```
cat /etc/shells
```

```
````
```

Ini akan menampilkan daftar shell yang tersedia di sistem.

## 2. **\*\*Ganti Shell Default\*\***:

```
```sh
chsh -s /bin/zsh
```
```

Gantilah `/bin/zsh` dengan path shell yang diinginkan dari output `/etc/shells`.

## 3. **\*\*Logout dan Login\*\***: Keluar dari sesi saat ini dan masuk kembali untuk melihat perubahan.

### #### Risiko dan Pertimbangan:

- **\*\*Kompatibilitas\*\***: Beberapa skrip atau aplikasi mungkin mengasumsikan penggunaan Bash dan tidak berfungsi dengan baik di shell lain.
- **\*\*Konfigurasi\*\***: Konfigurasi dan plugin mungkin perlu diatur ulang atau diadaptasi ke shell baru.
- **\*\*Sistem Skrip\*\***: Jika ada skrip sistem yang berjalan dengan asumsi menggunakan Bash, perlu dipastikan skrip tersebut kompatibel dengan shell baru atau dibiarkan tetap menggunakan Bash.

## ### 3. Perbedaan antara Mode Interaktif dan Non-Interaktif dalam Penggunaan Shell

### #### Mode Interaktif:

- **\*\*Penggunaan\*\***: Digunakan saat pengguna langsung mengetik perintah di prompt.
- **\*\*Fitur\*\***: Mendukung fitur seperti auto-completion, prompt kustom, dan auto-suggestions.
- **\*\*Eksekusi\*\***: Perintah dieksekusi segera setelah ditekan Enter.

### #### Mode Non-Interaktif:

- **\*\*Penggunaan\*\***: Digunakan saat menjalankan skrip atau batch jobs.

- **Fitur**: Tidak mendukung fitur interaktif, hanya mengeksekusi perintah yang ada dalam skrip.
- **Eksekusi**: Semua perintah dalam skrip dijalankan secara berurutan tanpa interaksi pengguna.

#### Contoh Kasus:

- **Interaktif**: Saat menggunakan shell untuk menavigasi sistem file atau menjalankan perintah ad-hoc.
- **Non-Interaktif**: Saat menjalankan cron job atau script deployment otomatis.

#### ### 4. Script Shell untuk Backup File

```
```sh
```

```
#!/bin/bash
```

```
# Meminta input dari pengguna
```

```
read -p "Masukkan nama file yang akan di-backup: " file
```

```
read -p "Masukkan lokasi tujuan backup: " backup_location
```

```
# Memeriksa apakah file ada
```

```
if [ ! -f "$file" ]; then
```

```
    echo "File tidak ditemukan!"
```

```
    exit 1
```

```
fi
```

```
# Memeriksa apakah direktori tujuan ada
```

```
if [ ! -d "$backup_location" ]; then
```

```
    echo "Lokasi tujuan tidak ditemukan!"
```

```
    exit 1
```

```
fi
```

```
# Melakukan backup file
```

```
cp "$file" "$backup_location"
```

```
# Memeriksa hasil operasi
```

```
if [ $? -eq 0 ]; then
```

```
    echo "Backup berhasil!"
```

```
else
```

```
    echo "Backup gagal!"
```

```
fi
```

```
```
```

### ### 5. Otomatisasi Tugas Administrasi Sistem dengan Script Shell

#### #### Contoh Tugas: Backup Harian Direktori

Script ini akan otomatis melakukan backup direktori `/home/user/data` ke `/backup` setiap hari.

#### #### Langkah-langkah dalam Script:

##### 1. Buat Script:

```
```sh
```

```
#!/bin/bash
```

```
# Direktori yang akan di-backup
```

```
SOURCE_DIR="/home/user/data"
```

```
BACKUP_DIR="/backup/$(date +%F)"
```

Membuat direktori backup dengan nama berdasarkan tanggal

```
mkdir -p "$BACKUP_DIR"
```

Melakukan backup

```
cp -r "$SOURCE_DIR"/* "$BACKUP_DIR/"
```

Memeriksa hasil operasi

```
if [ $? -eq 0 ]; then
```

```
    echo "Backup berhasil!"
```

```
else
```

```
    echo "Backup gagal!"
```

```
fi
```

```
---
```

2. **Jadwalkan dengan Cron**:

- Buka crontab:

```
``sh
```

```
crontab -e
```

```
---
```

- Tambahkan entri berikut untuk menjalankan script setiap hari pada jam 2 pagi:

```
``sh
```

```
0 2 * * * /path/to/backup_script.sh
```

```
---
```

Penjelasan:

- `mkdir -p "\$BACKUP_DIR"`: Membuat direktori backup dengan nama berdasarkan tanggal saat ini.

- ``cp -r "$SOURCE_DIR"/* "$BACKUP_DIR/"``: Menyalin seluruh konten dari direktori sumber ke direktori backup.

- Cron Job: Menjadwalkan script untuk berjalan otomatis pada jam 2 pagi setiap hari.

Dengan script seperti ini, tugas-tugas backup yang memerlukan waktu dan konsistensi bisa diotomatisasi, mengurangi risiko kehilangan data dan memastikan backup dilakukan secara teratur.

BAB II

KESIMPULAN

Kesimpulannya, perintah-perintah dalam shell Linux memungkinkan pengguna untuk melakukan berbagai tugas secara efisien. Penggunaan perintah seperti ``find`` untuk mencari file, ``echo`` untuk menulis atau menambahkan teks ke dalam file, serta ``cat``, ``more``, dan ``grep`` untuk menampilkan dan mencari isi file adalah dasar yang penting dalam pengelolaan file dan teks. Selain itu, penggunaan editor teks ``nano`` untuk membuat dan mengedit skrip shell seperti ``Latihan1.sh``, ``Latihan2.sh``, ``Latihan3.sh``, dan ``Latihan4.sh``, serta memberikan izin eksekusi dengan ``chmod +x`` dan menjalankannya, menunjukkan bagaimana skrip shell dapat digunakan untuk otomatisasi dan pengelolaan sistem. Mempelajari dan memahami perintah-perintah ini sangat berguna untuk meningkatkan produktivitas dan efisiensi dalam lingkungan Linux.