

SIGNALS

Definisi

- *Signal* adalah *small integer* yang biasanya kurang dari 32.
- *Signal* bisa dikirimkan oleh sebuah proses ke proses yang lain sebagai tanda bahwa sebuah event telah terjadi dan untuk melakukan interupsi ke proses yang dituju.
- **Prerequisite:** untuk mengirimkan sebuah *signal* ke sebuah proses, maka proses tersebut harus berjalan terlebih dahulu.

Signal Generation

- Munculnya *hardware exception*
- *Signal* di-generate salah satunya dari pengguna yang mengetik salah satu karakter khusus pada *keyboard*, misalnya *ctrl+g*
- Munculnya *software event*

Signal Actions

- Ketika sebuah proses menerima sebuah sinyal, maka proses tersebut bisa melakukan beberapa aksi dibawah ini:
 - Mengabaikan sinyal yang datang
 - Proses tersebut di-terminated (*killed*)
 - Meng-generate sebuah file *core dump*
 - Proses tersebut dimatikan
 - Eksekusi sebuah proses dilanjutkan setelah sebelumnya dihentikan

Berikut tabel dari Signals:

Signal	Value	Action	Comment
SIGHUP	1	Term	Hangup detected on controlling terminal or death of controlling process
SIGINT	2	Term	Interrupt from keyboard
SIGQUIT	3	Core	Quit from keyboard
SIGILL	4	Core	Illegal Instruction
SIGABRT	6	Core	Abort signal from abort(3)
SIGFPE	8	Core	Floating point exception
SIGKILL	9	Term	Kill signal
SIGSEGV	11	Core	Invalid memory reference
SIGPIPE	13	Term	Broken pipe: write to pipe with no readers

TERM=terminate, CORE=terminate and dump

- **SIGALRM** 14 Term Timer signal from alarm(2)
- **SIGTERM** 15 Term Termination signal
- **SIGUSR1** 30,10,16 Term User-defined signal 1
- **SIGUSR2** 31,12,17 Term User-defined signal 2
- **SIGCHLD** 20,17,18 Ign Child stopped or terminated
- **SIGCONT** 19,18,25 Cont Continue if stopped
- **SIGSTOP** 17,19,23 Stop Stop process
- **SIGTSTP** 18,20,24 Stop Stop typed at tty
- **SIGTTIN** 21,21,26 Stop tty input for background process
- **SIGTTOU** 22,22,27 Stop tty output for background process
- The signals **SIGKILL** and **SIGSTOP** cannot be caught, blocked, or ignored.
- **IGN** : Ignore Signal, **STOP** : Stop the Process

Adapun, *signals* yang paling sering digunakan adalah

Name	Number	Meaning
HUP	1	Hang Up. The controlling terminal has gone away.
INT	2	Interrupt. The user has pressed the interrupt key (usually Ctrl-C or DEL).
QUIT	3	Quit. The user has pressed the quit key (usually Ctrl-\\). Exit and dump core.
KILL	9	Kill. This signal cannot be caught or ignored. Unconditionally fatal. No cleanup possible.
TERM	15	Terminate. This is the default signal sent by the kill command.
EXIT	0	Not really a signal. In a shell script, an EXIT trap is run on any exit, signalled or not.

Contoh Implementasi Signal pada Script Sederhana

1. Buat sebuah *script* dengan nama trap.sh dan isi dengan kode dibawah ini:

```
#!/bin/bash

trap "echo Hello World" USR2
while true
do
    echo Running....
    sleep 2
done
```

2. Penjelasan script:

```
#!/bin/bash

trap "echo Hello World" USR2
```

Artinya, kita mendefinisikan **trap** dengan *signal* USR2, sehingga setiap kali *signal* itu diterima maka perintah “echo Hello World” akan di eksekusi

```
while true
do
    echo Running....
    sleep 2
done
```

While loop biasa, yang akan selalu menjalankan perintah **echo Running....** selamanya.

3. Jadikan *script* tersebut bisa dieksekusi, dengan menulis *command* **chmod +x trap.sh**
4. Jalankan **script** dengan *command* **./trap.sh** di salah satu terminal
Output ketika *script* tersebut dijalankan adalah seperti ini:

```
user@sysprog-ova:~/signal$ ./trap.sh
Running....
Running....
Running....
Running....
Running....
Running....
Running....
Running....
Running....
Running....
Running....
Running....
Running....
Running....
Running....
Running....
Running....
```

5. Buka terminal yang lain, lalu cek PID dari *process* eksekusi *script* trap.sh dengan *command* **ps -ef | grep trap.sh**
Berikut adalah outputnya

```
user@sysprog-ova:~/signal$ ps -ef | grep trap.sh
user  2094  1910  0 04:01 pts/0    00:00:00 /bin/bash ./trap.sh
user  2112  2051  0 04:02 pts/1    00:00:00 grep --color=auto trap.sh
```



PID Process

6. Setelah kita mendapatkan PID dari *process* tersebut, kita bisa mengirimkan *signal* untuk *process* yang sedang berlangsung. Misalnya dengan mengirimkan *signal* yang sudah kita *define* sebelumnya yaitu USR2. **Kill -[SIGNAL] [PID]**

```
user@sysprog-ova:~/signal$ kill -USR2 2094
```

Maka, output pada terminal sebelumnya menjadi seperti ini:

A terminal window titled 'user@sysprog-ova: ~/signal'. The output shows a repeating pattern of 'Running....' followed by 'Hello World'. The text is as follows:

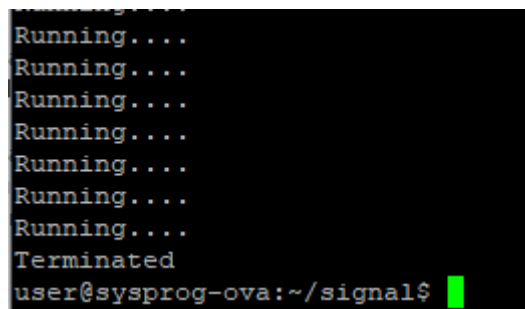
```
Running....
Running....
Running....
Hello World
Running....
Running....
Running....
Running....
```

Yaitu, terselip kata “Hello World” akibat eksekusi “echo Hello World” yang diaktifkan oleh *signal* USR2.

7. Karena program tersebut menjalankan *infinite loop*, maka kita bisa mencoba mengirimkan sinyal untuk meng-*terminate* proses yang sedang terjadi. Yaitu dengan menjalankan perintah dibawah ini. Karena TERM sebelumnya tidak di *define*, maka TERM memakai definisi *default*, yaitu meng-*terminate* sebuah proses.

```
user@sysprog-ova:~/signal$ kill -TERM 2094
```

Maka,

A terminal window showing the continuation of the previous output. It shows several 'Running....' lines, followed by 'Terminated', and then the prompt 'user@sysprog-ova:~/signal\$' with a green cursor. The text is as follows:

```
Running....
Running....
Running....
Running....
Running....
Running....
Running....
Terminated
user@sysprog-ova:~/signal$
```

Process tersebut akan *terminated*.