Makefile

Daftar Isi

Apa itu Makefile?

Makefile berfungsi untuk memerintahkan compile dan link sebuah program.

Manual Book

- GNU Make (PDF)
- GNU Make (Web)

Install Make

Windows

- Download and Install Make
- add Make to path

Syntax Dasar

Syntax dasar

```
target: prerequisities
<TAB> recipe
```

target adalah nama fungsi dan *prerequisities* adalah yang mengikuti *target*. *Command*-nya disebut *recipe*. *Recipe* menggunakan *prerequisites* untuk membuat *target*. *target*, *prerequisites*, dan *recipe* membentuk sebuah *rule*.

Single Command

Sebagai contoh, buatlah sebuah file tanpa *extension* dengan nama Makefile.

```
say_hello:
     echo "Hello World"
```

Jalankan di terminal dengan perintah *make*, hasilnya:

```
$ make
echo "Hello Wolrd"
Hello World
```

Hasil di atas menampilkan *command* dan hasilnya. *Command* dapat disembunyikan dengan menambahkan @ di depan **echo**.

```
say_hello:
   @echo "Hello World"
```

Hasilnya adalah:

```
$ make
Hello World
```

Lebih dari Satu Commands

Apabila terdapat lebih dari 1 *target*, dengan menggunakan *syntax* di bawah ini maka hanya *target* yang ditulis pertama kali yang akan dijalankan.

```
cmd01:
    @echo "command ke-1"

cmd02:
    @echo "command ke-2"
```

Agar semua *command* dijalankan maka perlu ditambah *all* sebagai berikut:

```
all: cmd01 cmd02

cmd01:
    @echo "command ke-1"

cmd02:
    @echo "command ke-2"
```

Tambahkan juga spesial *phony target* untuk mendefinisikan bahwa semua *target* adalah bukan *files*. Ini agar tidak membuat bingung *make* dalam menjalankan *target*-nya. Berikut ini adalah contoh penulisan *phony*:

```
.PHONY: all cmd01 cmd02

all: cmd01 cmd02

cmd01:
    @echo "command ke-1"

cmd02:
    @echo "command ke-2"
```

Referensi

- Opensource: what is a Makefile and how does it work?

Variables & Pattern Rules

Variable

Variable berguna agar sebuah command dapat dengan mudah diganti dengan command yang lainnya. Misalnya:

```
CC        := gcc
```

Automatic variables

Berguna untuk mengganti nama dari target atau prerequisite.

- $@: diganti dengan nama target
- $<: diganti dengan nama pertama dari prerequisite
- @^: diganti dengan nama semua prerequisite

Pattern rules

```
%html: %.rst
        <recipe>
```

Rule di atas untuk meng-compile sebuah file dengan akhiran html dari file rst.

Contoh Penggunaan

Berikut ini contoh penggunaan pattern rule untuk mengkonversi file rst ke html dengan menggunakan rst2html.py.

```
%.html: %.rst
        rst2html.py $< $@
```

Misalnya di dalam direktori tersebut terdapat file rst yang bernama latihan.rst. Kemudian jalankanlah:

```
make latihan.html
```

Hasilnya pada direktori tersebut akan di-generate sebuah file dengan nama latihan.html.

Wildcard

Definisi

Wildcard berfungsi untuk mendaftar semua file dengan ekstensi yang didefinisikan.

Contoh

```
SRC:= $(wildcard *.rst)
OUT:= $(SRC:%.rst=%.html)

all: $(OUT)

%.html: %.rst
        rst2html.py $< $@
```

Penjelasan syntax

- `SRC:= $(wildcard *.rst)` > mendaftar nama semua file yang bearkhiran .rst
- `OUT:= $(SRC:%.rst=%.html)` > mengganti file yang berakhiran rst dengan html
- `all: $(OUT)` > untuk memanggil rule
- `%.html: %.rst` > ketika daftar nama di $(OUT) match, maka recipe akan dijalankan