Politechnika Wrocławska

Programowanie Systemowe

LABORATORIUM

Temat: **Pliki Makefile**

Hanna Kieszek – 283797

26.10.2025

1. Cel ćwiczenia

Zapoznanie się z plikami Makefile i ich możliwościami.

2. Wykorzystane narzędzia

1. Debian
2. Terminal

3. Przebieg laboratorium

3.1. Czym jest plik Makefile i jego utworzenie

Makefile jest specjalnym plikiem tekstowym który zawiera instrukcje dotyczące operacji na plikach źródłowych. Działa on na tej zasadzie, że program make odczytuje plik Makefile i wykonuje reguły w nim zawarte. Reguła to pojedyncza instrukcja opisująca jedno konkretne zadanie.

Na początku został utworzony plik Makefile za pomocą komendy ***touch Makefile*** a następnie otworzony przy pomocy edytora tekstu nano.

3.2. Tworzenie pierwszych reguł

Pierwsza reguła jaka została utworzona to ***all.*** Jest to reguła którą wywołujemy poprzez wpisanie w terminalu ***make***, ponieważ znajduje się ona tam jako pierwsza. Po wywołaniu jej zostaną wykonane instrukcje które znajdują się w regule all. Drugą regułą jest clean która odpowiada za usuwanie obiektów.

Celem pierwszej części zadania było napisanie reguły która będzie tworzyć pliku który w swojej nazwie będzie zawierał bieżącego użytkownika i timestamp oddzielone kropką oraz drugiej reguły która będzie ten plik usuwać. Aby uzyskać obecnego użytkownika i obecny timestamp zostały użyte zmienne globalne z odpowiednimi poleceniami:

```
timestamp=$(shell date +%s)
user=$(shell whoami)
```

Następnie została utworzona reguła all:

```
all:
	touch $(user).$(timestamp)
```

Oraz reguła clean:

```
clean:
	rm  $(user).*
```

Ponieważ dokładny timestamp tworzonego pliku nie jest znany, gdyż jest on generowany w momencie wywołania make , została użyta konstrukcja $(user).* gdzie * zastępuje resztę nazwy pliku. Po wywołaniu make w terminalu zostanie utworzony plik user.timestamp natomiast kiedy wywołamy clean zostanie on usunięty.

3.3. Tworzenie reguły data.txt

Zadaniem reguły data.txt będzie zapis słowa “data” do pliku data.txt. Wygląda ona w ten sposób:

```
data.txt:
	echo data  >  data.txt
```

Taka konstrukcja sprawia, że jeśli plik data.txt nie istnieje to zostanie on utworzony, natomiast jeżeli istnieje to jego zawartość zostanie nadpisana.

3.4. Modyfikowanie pliku Makefile i dodanie zależności

Na tym etapie plik Makefile wygląda w ten sposób:

```
timestamp=$(shell date +%s)
user=$(shell whoami)

all:
	touch $(user).$(timestamp)

clean:
	rm  $(user).*

data.txt:
	echo data > data.txt
```

Celem kolejnej części zadania było zmodyfikowanie go w taki sposób aby w regule all był tworzony katalog o nazwie podanej w zmiennej globalnej OUTDIR. Oprócz tego plik $(user).$(timestamp) będzie zapisywany w tym folderze a jego treść będzie skopiowana z pliku data.txt. Poniżej znajduje się wygląd nowej reguły all oraz zmiennych globalnych:

```
timestamp=$(shell date +%s)
user=$(shell whoami)
OUTDIR=build
```

```
all: data.txt
	mkdir -p $(OUTDIR)
	cp data.txt  $(OUTDIR)/$(user).$(timestamp)
```

Komenda na tworzenie folderu mkdir zawiera opcję -p dzięki której nie będzie wyskakiwał błąd jeżeli taki katalog już istnieje.
Aby zabezpieczyć się przed próbą kopiowania pliku który nie istnieje została dodana zależność pomiędzy all a data.txt (all: data.txt). Dzięki temu zanim make wykona instrukcje zawarte w all, sprawdzi czy data.txt istnieje. Jeśli nie istnieje, reguła data.txt zostanie wykonana automatycznie.

Zmieniona reguła clean będzie odpowiadała za usuwanie folderu OUTDIR oraz pliku data.txt, a komenda w niej zawarta wygląda tak:

```
clean:
	rm -rf $(OUTDIR) data.txt
```

Opcja -rf oznacza, że katalog będzie usunięty z całą jego zawartością oraz wymusza usunięcie bez pytania o potwierdzenie oraz ignoruje błąd jeżeli nie znajdzie określonego w zapytaniu pliku.

Następnie w regule data.txt zamieniona została zawartość pliku date.txt na obecny timestamp.

```
data.txt:
	echo $(timestamp)  > data.txt
```

Finalnie plik Makefile ma następującą formę:

```
timestamp=$(shell date +%s)
user=$(shell whoami)
OUTDIR=build

all: data.txt
	mkdir -p $(OUTDIR)
	cp data.txt  $(OUTDIR)/$(user).$(timestamp)

clean:
	rm -rf $(OUTDIR) data.txt

data.txt:
	echo $(timestamp)  > data.txt
```