Pavel Rychlý

Makefile, Make, Mk

how to use them for data processing

Make

- traditionally for building binary programs from sources
- C, C++, Fortran

Make

- traditionally for building binary programs from sources
- C, C++, Fortran
- aa.h, bb.h, aa.c, bb.c, main.c

Make

- traditionally for building binary programs from sources
- C, C++, Fortran
- aa.h, bb.h, aa.c, bb.c, main.c
- create aa.o, bb.o (binary objects), ab.a (library)
- main (runtime binary)
- handling dependencies

Makefile

- declaration of dependencies
- specification of rules
  - for concrete target (`main` from `main.o, ab.a`)
  - generic (from *.c to *.o)
  - many defaults

Makefile for data

- it is better to process data in steps
- corpus: html – prevert – vert – annotated
- it could be in one pipeline (at the end)
- but we want to see partial results for debugging during development

Makefile for data

- corpus: html – prevert – vert – annotated
- from html to pre-vertical: html2prevert.py

```
%.prev: %.html
        html2prevert.py <$< >$@

%.vert: %.prev
        tokenize $< >$@

%.tags: %.vert
        desamb.sh <$< >$@
```

Makefile for data

- corpus: html – prevert – vert – annotated
- from html to pre-vertical: html2prevert.py

```
%.prev: %.html
    html2prevert.py -skip-h -m 20 -stopw /nlp/cor... <$< >$@

%.vert: %.prev
    sed -e 's/\([0-9]\) -/\1-/g' $< | tokenize |grep -v '^_' >$@

%.tags: %.vert
    desamb-utf8-majka.sh -skipdis <$< | sed -e 's/^@.*/@\tk4' >$@
```

Makefile

- configuration options in variables

```
MAJKA=/nlp/projekty/ajka/bin/majka
%.annot: %.vert
    $(MAJKA) -p <$< >$@
```

- list of files/targets

```
PREFS=4 5 6 7 8 9 $(shell seq -w 00 17)
DIRS=$(wildcard SPACE14/20??)

corps: $(DIRS:%=%.cvert)

%.cvert: $(PREFS:%=\%/%.vert)
cat $^ >$@
```

- variables from commandline: make PREFS='1 2 3'

Make

- run in parallel: make -j 8
- run in max load: make -l [load]
- dry run: make -n
- remake all: make -B

problems Make

- TAB at the line beginning
- each line run in separate shell invocation
- escape all $

Mk

- from last version of Unix, first version of Plan9
- no TABS
- simple substitution rules
- dependencies using command

More info

- Book: *Managing Projects with GNU Make* (by Robert Mecklenburg)
- GNU Make https://www.gnu.org/software/make/manual/
- Totorial: Automation and Make https://swcarpentry.github.io/make-novice/