Makefiles

Jens Dalsgaard Nielsen

```
FILE     := files
DOCS :=$(file < $(FILE))

# no need for = use  :=  here – why ?
VERSION = "V-220531"
HDOCS=$(addsuffix .html, $(DOCS))
PHDOCS=$(HDOCS)
```

run commands and save result

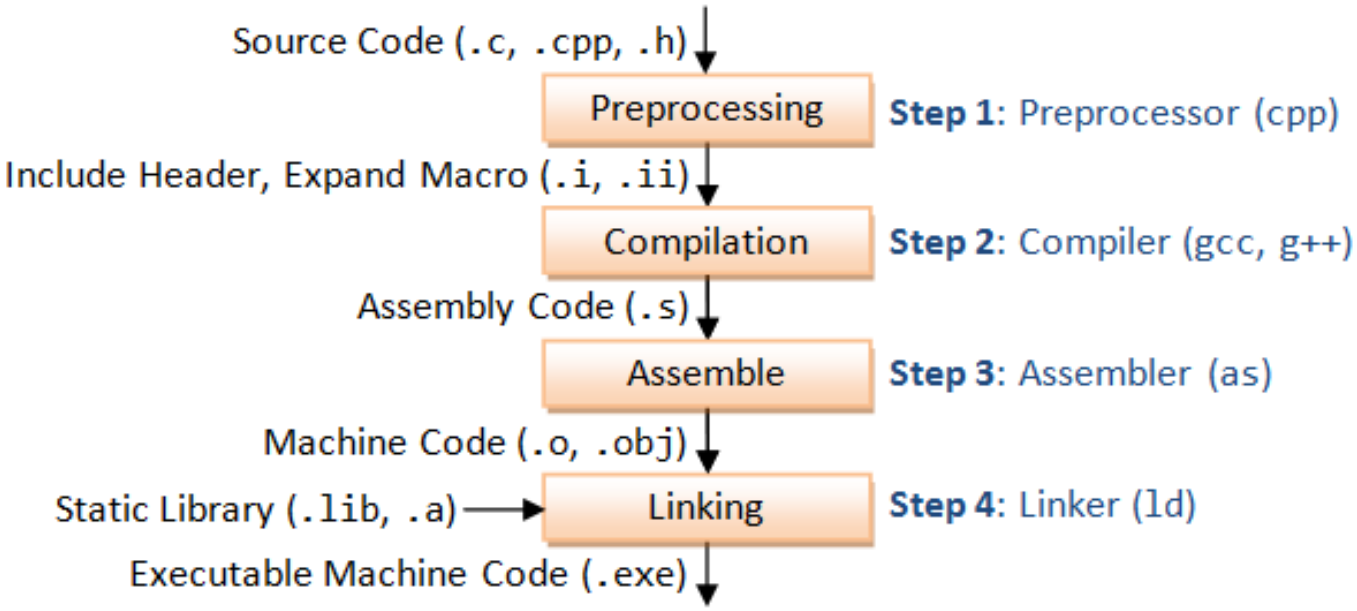
```
MAAU:="js1.es.aau.dk"
MJENSD:="jensd.dk@linux94.unoeuro.com"

# run commands
LOC:=`pwd | sed 's(.*www(public_html(''`
LOCJ:=`pwd | sed 's(.*www/jensd(public_html(''`
STR:="/*"
LOCC:=$(LOC)$(STR)
LOCDC:=$(LOCJ)$(STR)


# indrykning nedenunder SKAL være en tab og altså IKKE spaces
.PHONY : jensd
jensd:
   rm -rf *~
   chmod -fR 0700 *jemdoc
   chmod -fR 0700 *py
   cp -f jensdlogo.png logo.png
   sshpass -f ref-to-file-passwd     rsync -lut --exclude '*~' * $(MJENSD):$(LOC)/

--- make jensd  :-)
```

compilerens way to exec



Automatic Variables

Automatic variables are set by make after a rule is matched. There include:

```
    $@: the target filename.
    $*: the target filename without the file extension.
    $<: the first prerequisite(forudsætning) filename.
    $^: the filenames of all the prerequisites, separated by spaces, discard duplicates.
    $+: similar to $^, but includes duplicates.
    $?: the names of all prerequisites that are newer than the target, separated by spaces.
```

For example, we can rewrite the earlier makefile as:

```
all: hello

# $@ matches the target; $< matches the first dependent
  |----------- target
  |      |--- first dependent
hello: hello.o
    gcc -o $@ $<

hello.o: hello.c
   gcc -c $<

clean:
   rm hello.o hello.exe
```

se også https://www3.ntu.edu.sg/home/ehchua/programming/cpp/gcc_make.html

run python program if file of type jemdoc is newer than html

```
% : wildcard

%.html : %.jemdoc MENU
    sed 's/HHRR/~~~\n{}{raw}\n<hr>\n~~~/' < $<  > xxx
    python2.7 ./jemdoc.py  -c jdn.conf -o $@ xxx
    rm -f xxx
```

- A simple assignment
  - A simple assignment expression is evaluated only once, at the very first occurrence. For example, if CC :=${GCC} ${FLAGS} during the first encounter is evaluated to gcc -W then each time ${CC} occurs it will be replaced with gcc -W.
- Recursive assignment =
  - A Recursive assignment expression is evaluated everytime the variable is encountered in the code. For example, a statement like CC = ${GCC} {FLAGS} will be evaluated only when an action like ${CC} file.c is executed. However, if the variable GCC is reassigned i.e GCC=c++ then the ${CC} will be converted to c++ -W after the reassignment.