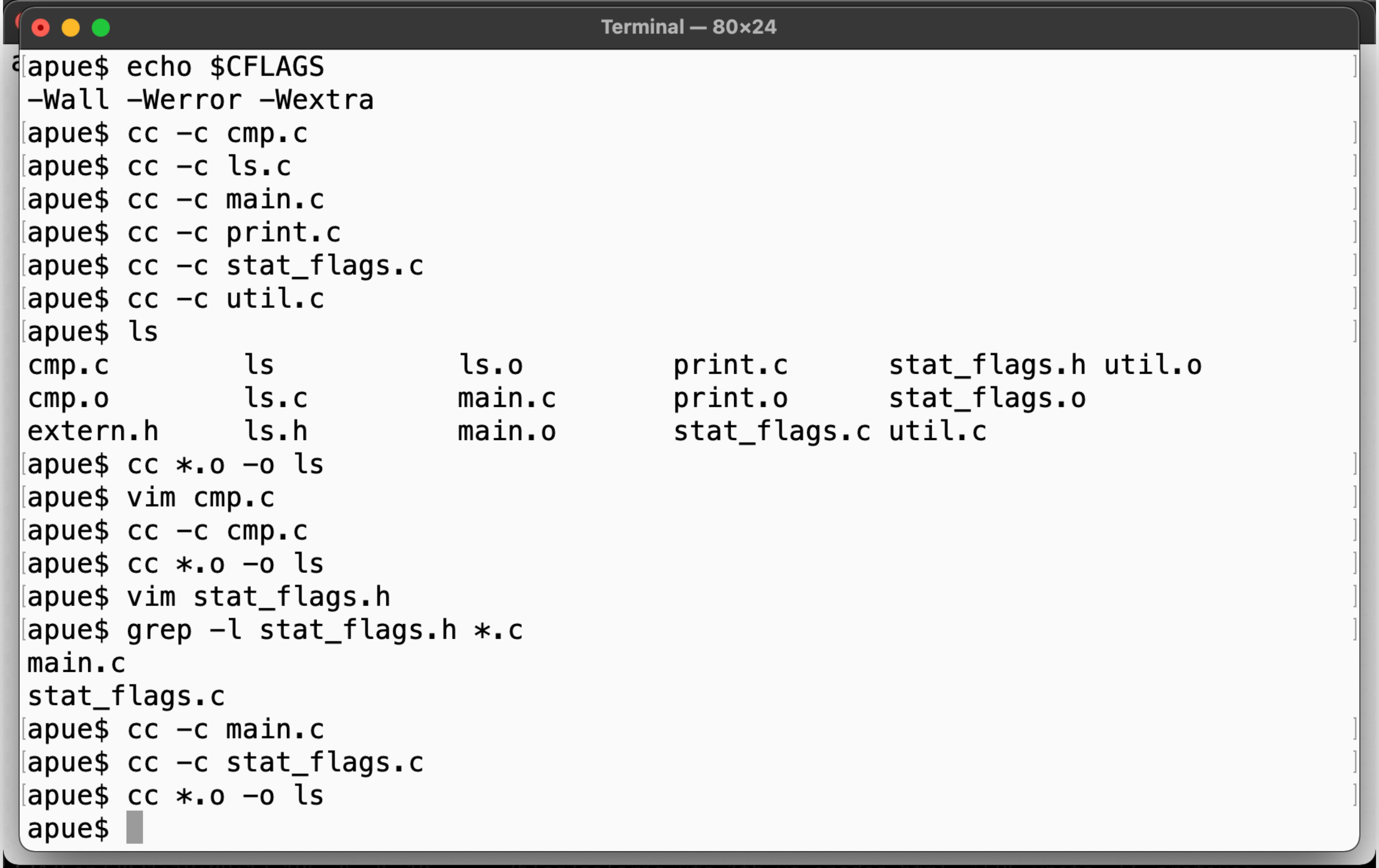
Advanced Programming in the UNIX Environment

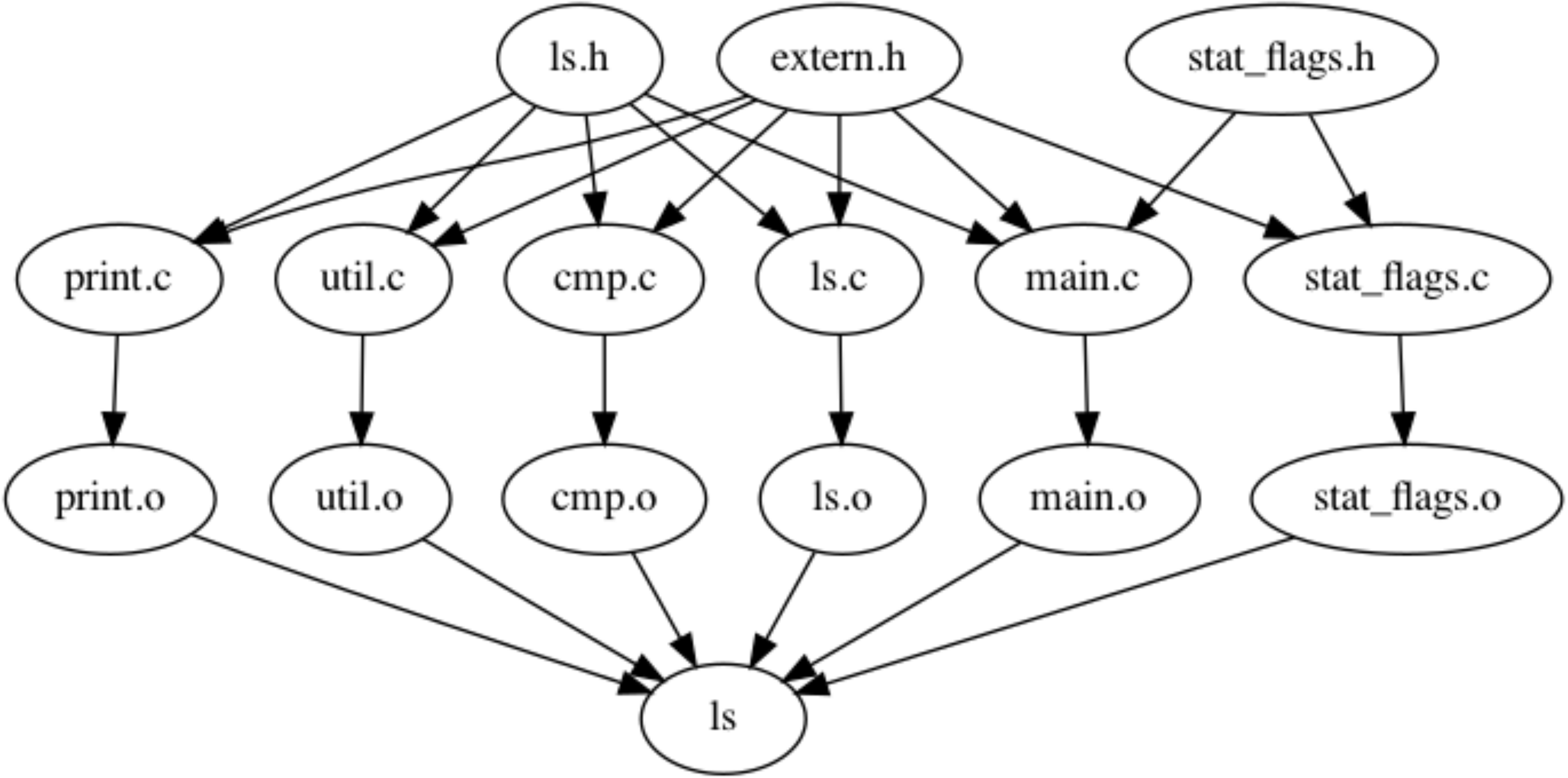
Week 05, Segment 7: Unix Development Tools: `make(1)`

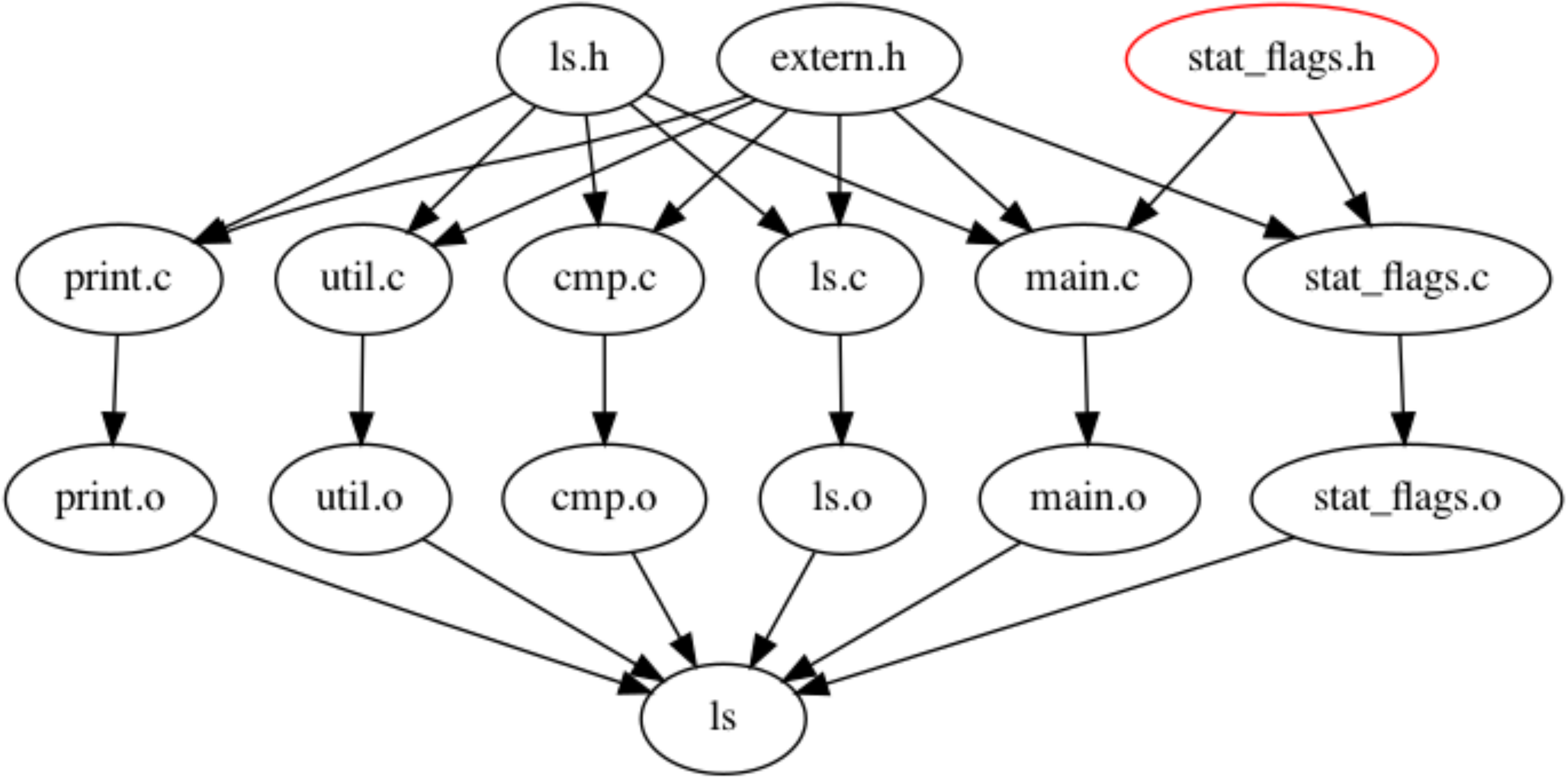
Department of Computer Science
Stevens Institute of Technology

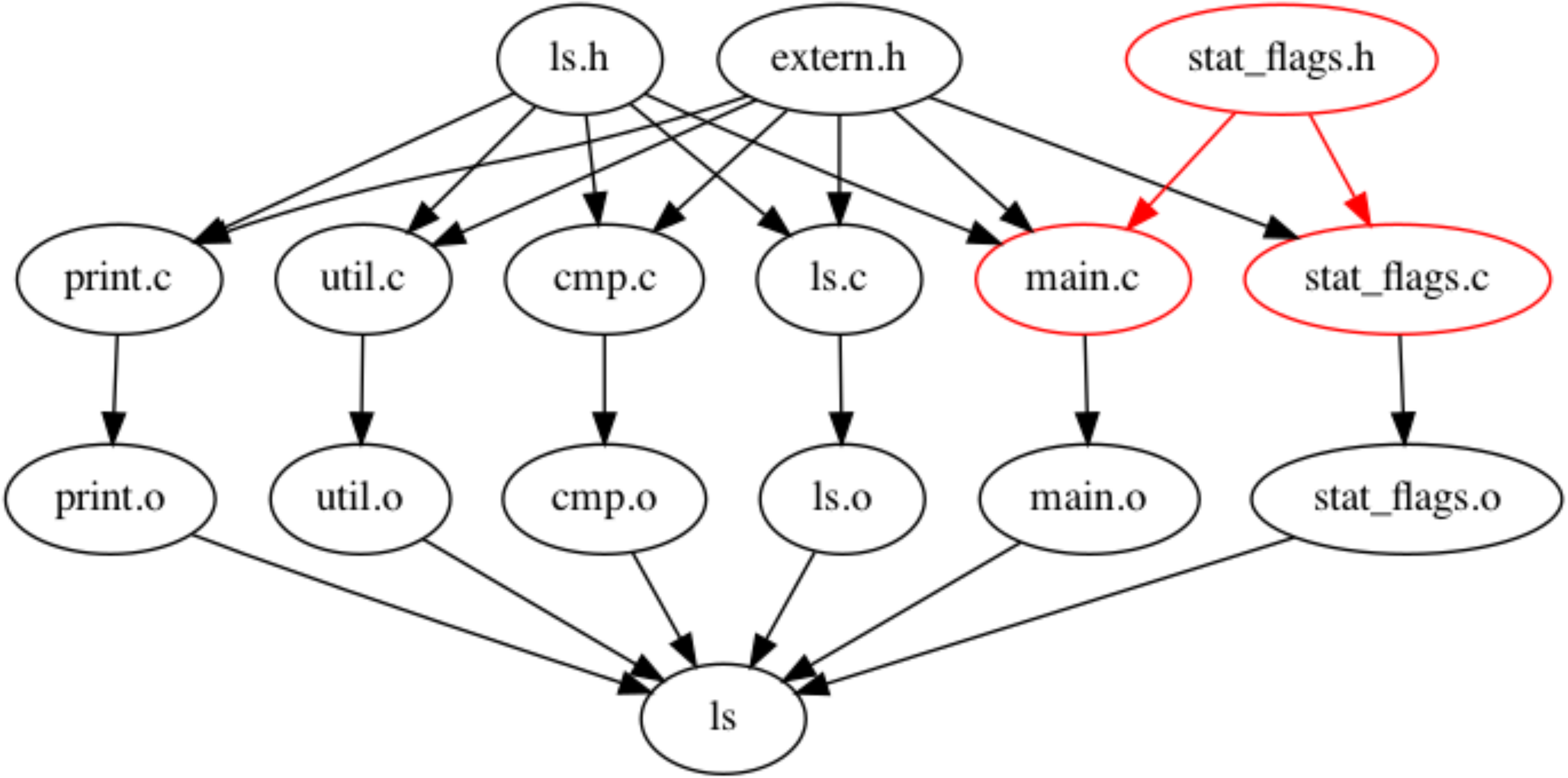
Jan Schaumann
jschauma@stevens.edu
https://stevens.netmeister.org/631/



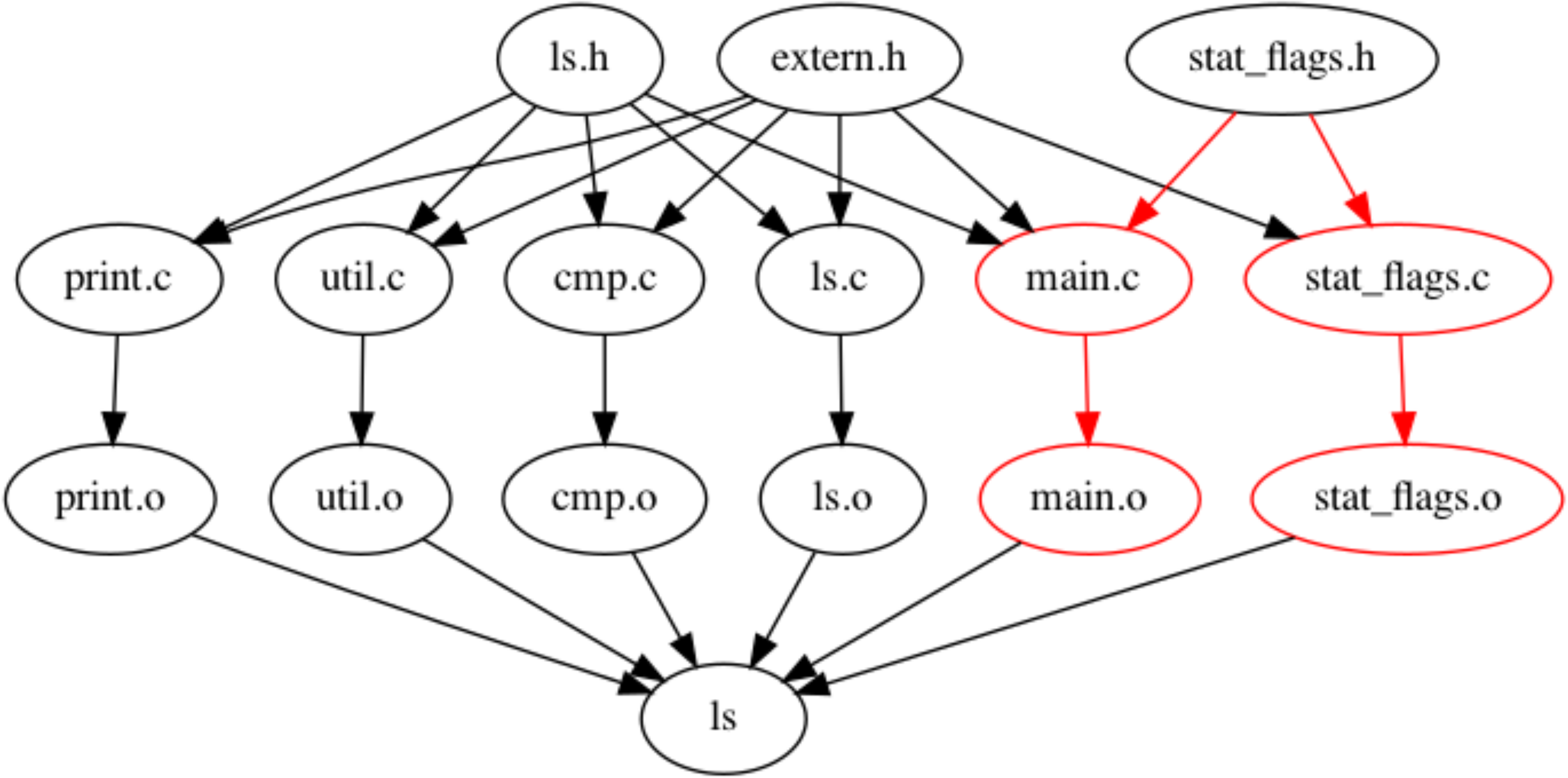
```
apue$ echo $CFLAGS
-Wall -Werror -Wextra
apue$ cc -c cmp.c
apue$ cc -c ls.c
apue$ cc -c main.c
apue$ cc -c print.c
apue$ cc -c stat_flags.c
apue$ cc -c util.c
apue$ ls
cmp.c        ls           ls.o         print.c      stat_flags.h util.o
cmp.o        ls.c         main.c       print.o      stat_flags.o
extern.h     ls.h         main.o       stat_flags.c util.c
apue$ cc *.o -o ls
apue$ vim cmp.c
apue$ cc -c cmp.c
apue$ cc *.o -o ls
apue$ vim stat_flags.h
apue$ grep -l stat_flags.h *.c
main.c
stat_flags.c
apue$ cc -c main.c
apue$ cc -c stat_flags.c
apue$ cc *.o -o ls
apue$ 
```

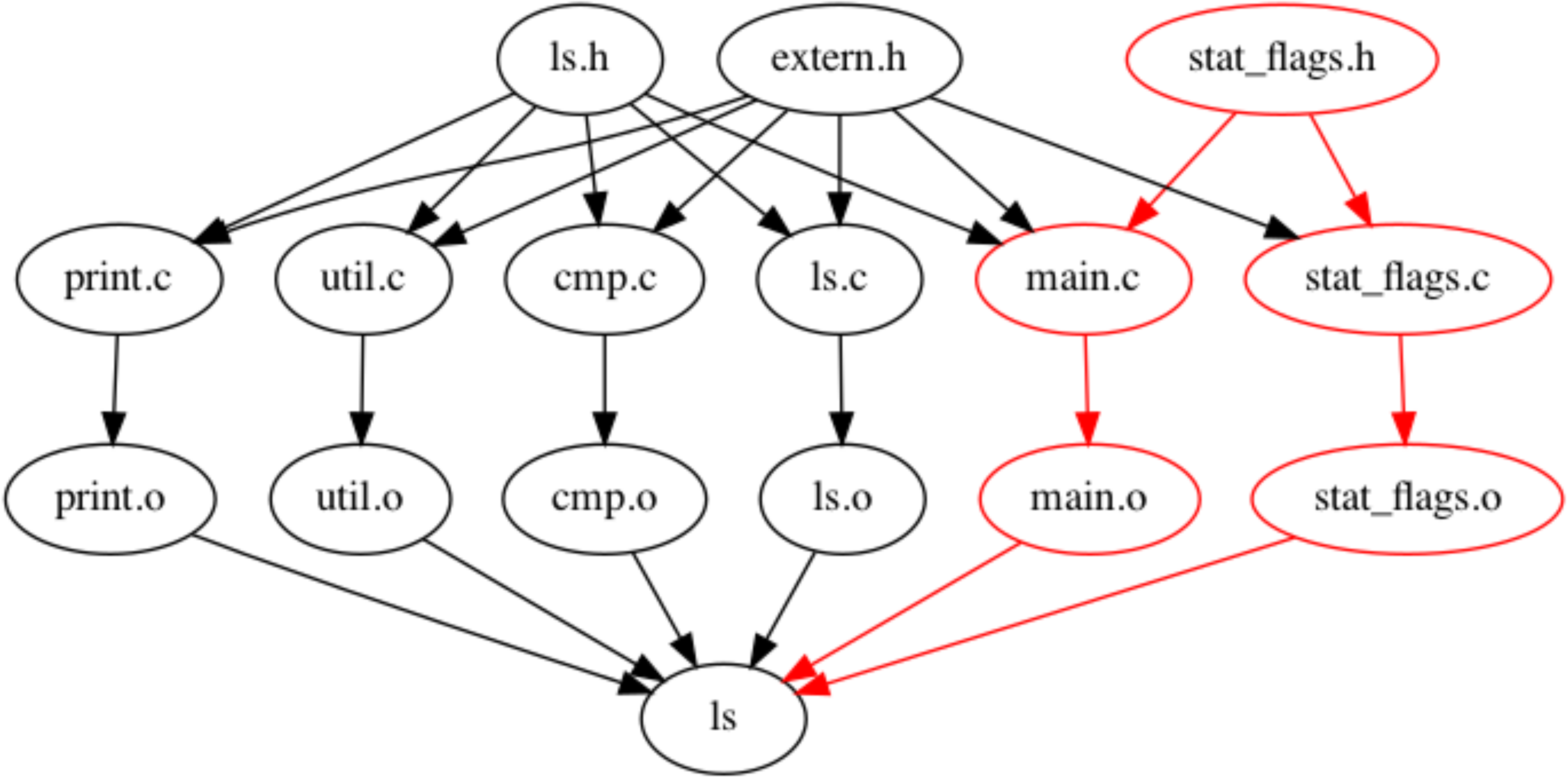

ls.h
extern.h
stat_flags.h
print.c
util.c
cmp.c
ls.c
main.c
stat_flags.c
print.o
util.o
cmp.o
ls.o
main.o
stat_flags.o
ls

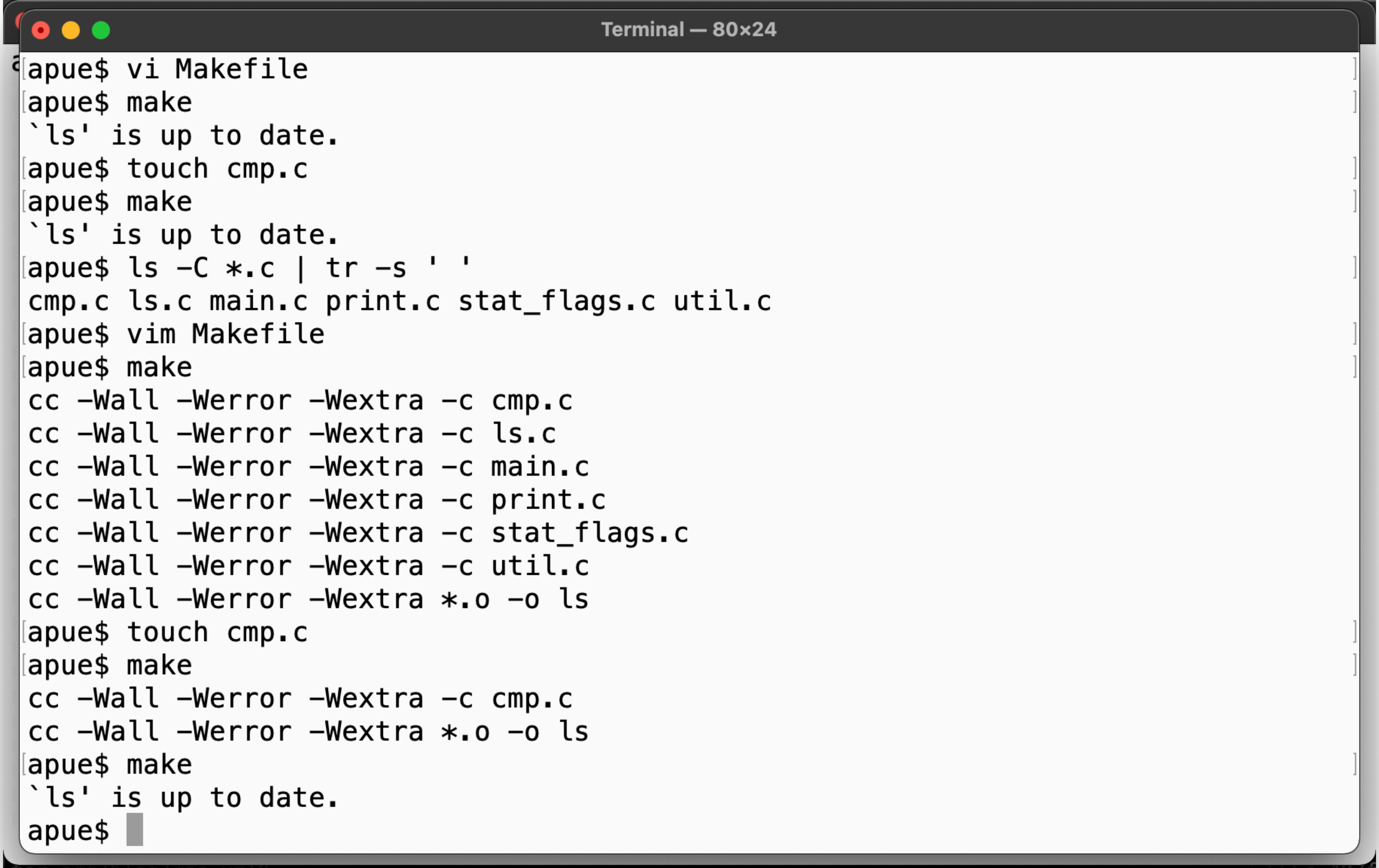

ls.h
extern.h
stat_flags.h
print.c
util.c
cmp.c
ls.c
main.c
stat_flags.c
print.o
util.o
cmp.o
ls.o
main.o
stat_flags.o
ls



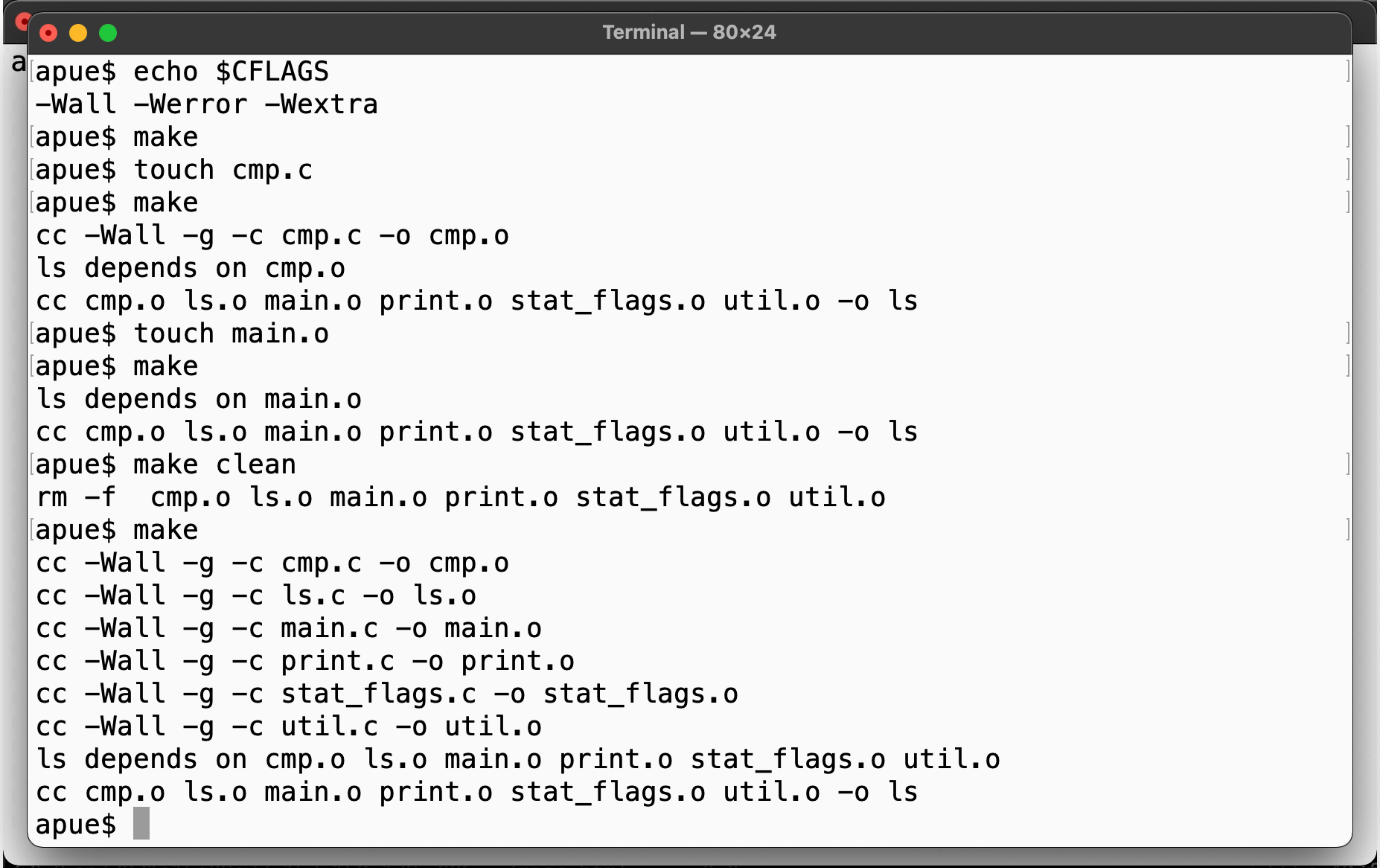
ls.h
extern.h
stat_flags.h
print.c
util.c
cmp.c
ls.c
main.c
stat_flags.c
print.o
util.o
cmp.o
ls.o
main.o
stat_flags.o
ls


ls.h
extern.h
stat_flags.h
print.c
util.c
cmp.c
ls.c
main.c
stat_flags.c
print.o
util.o
cmp.o
ls.o
main.o
stat_flags.o
ls

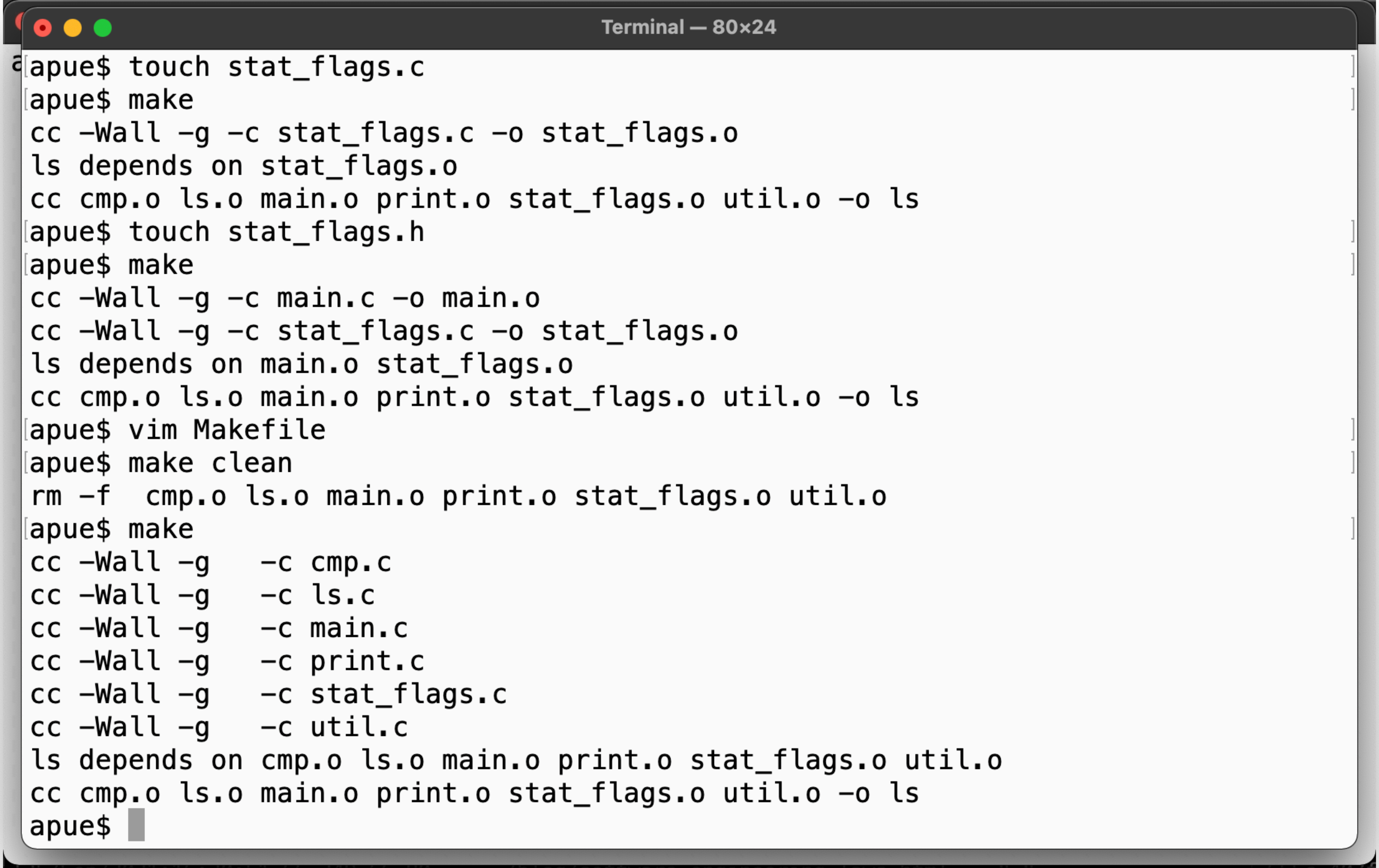

ls.h
extern.h
stat_flags.h
print.c
util.c
cmp.c
ls.c
main.c
stat_flags.c
print.o
util.o
cmp.o
ls.o
main.o
stat_flags.o
ls



```
apue$ vi Makefile
apue$ make
`ls' is up to date.
apue$ touch cmp.c
apue$ make
`ls' is up to date.
apue$ ls -C *.c | tr -s ' '
cmp.c ls.c main.c print.c stat_flags.c util.c
apue$ vim Makefile
apue$ make
cc -Wall -Werror -Wextra -c cmp.c
cc -Wall -Werror -Wextra -c ls.c
cc -Wall -Werror -Wextra -c main.c
cc -Wall -Werror -Wextra -c print.c
cc -Wall -Werror -Wextra -c stat_flags.c
cc -Wall -Werror -Wextra -c util.c
cc -Wall -Werror -Wextra *.o -o ls
apue$ touch cmp.c
apue$ make
cc -Wall -Werror -Wextra -c cmp.c
cc -Wall -Werror -Wextra *.o -o ls
apue$ make
`ls' is up to date.
apue$ 
```



```
apue$ echo $CFLAGS
-Wall -Werror -Wextra
apue$ make
apue$ touch cmp.c
apue$ make
cc -Wall -g -c cmp.c -o cmp.o
ls depends on cmp.o
cc cmp.o ls.o main.o print.o stat_flags.o util.o -o ls
apue$ touch main.o
apue$ make
ls depends on main.o
cc cmp.o ls.o main.o print.o stat_flags.o util.o -o ls
apue$ make clean
rm -f  cmp.o ls.o main.o print.o stat_flags.o util.o
apue$ make
cc -Wall -g -c cmp.c -o cmp.o
cc -Wall -g -c ls.c -o ls.o
cc -Wall -g -c main.c -o main.o
cc -Wall -g -c print.c -o print.o
cc -Wall -g -c stat_flags.c -o stat_flags.o
cc -Wall -g -c util.c -o util.o
ls depends on cmp.o ls.o main.o print.o stat_flags.o util.o
cc cmp.o ls.o main.o print.o stat_flags.o util.o -o ls
apue$
```



```
apue$ touch stat_flags.c
apue$ make
cc -Wall -g -c stat_flags.c -o stat_flags.o
ls depends on stat_flags.o
cc cmp.o ls.o main.o print.o stat_flags.o util.o -o ls
apue$ touch stat_flags.h
apue$ make
cc -Wall -g -c main.c -o main.o
cc -Wall -g -c stat_flags.c -o stat_flags.o
ls depends on main.o stat_flags.o
cc cmp.o ls.o main.o print.o stat_flags.o util.o -o ls
apue$ vim Makefile
apue$ make clean
rm -f  cmp.o ls.o main.o print.o stat_flags.o util.o
apue$ make
cc -Wall -g   -c cmp.c
cc -Wall -g   -c ls.c
cc -Wall -g   -c main.c
cc -Wall -g   -c print.c
cc -Wall -g   -c stat_flags.c
cc -Wall -g   -c util.c
ls depends on cmp.o ls.o main.o print.o stat_flags.o util.o
cc cmp.o ls.o main.o print.o stat_flags.o util.o -o ls
apue$ 
```

`make(1)`

`make(1)` is a command generator and build utility. Using a description file (usually `Makefile`) it creates a sequence of commands for execution by the shell.

- used to sort out dependency relations among files
- avoids having to rebuild the entire project after modification of a single source file
- performs *selective* rebuilds following a *dependency graph*
- allows simplification of rules through use of *macros* and *suffixes*, some of which are internally defined
- different versions of `make(1)` (BSD make, GNU make, Sys V make, ...) may differ in e.g.:
    - variable assignment and expansion/substitution
    - including other files
    - flow control (for-loops, conditionals etc.)