MP6

• This MP is simple:
  – Create a ‘make’ utility.
MP6

• This MP is simple:
  – Create a ‘make’ utility.

• What does ‘make’ do?
  – Reads a ‘makefile’
  – Determines the tasks that are available to run based on dependency rules
  – Run until all tasks are finished
MP6

job1: job2 job3
  commandtoberun withargs
  commandtoberun2 withargs

job2:
  othercommand

job3:
  finalcommand
target

job1: job2 job3
commandtoberun withargs
commandtoberun2 withargs

job2:
othercommand

job3:
finalcommand
job1: job2 job3

commandtoberun withargs
commandtoberun2 withargs

job2:
othercommand

job3:
finalcommand
MP6

job1: job2 job3
commandtoberun withargs
commandtoberun2 withargs

job2:
othercommand

job3:
finalcommand
MP6

- We can show this graphically:

...job1 depends on job2 and job3 being done.
In MP6, you will specify (with the –j # option) how many worker threads should run.
- “-j 1”: Only one worker thread
- “-j 2”: Two worker threads
- “-j 100”: One hundred worker threads

Use getopt() to handle command-line options
If the makefile is ran with \(-j\) 2, then:

[thread a]: job2 runs
[thread b]: job3 runs
[thread b]: job3 finishes
[thread b]: idle, job1 not ready
[thread a]: job2 finishes
[thread a OR b]: job1 runs
[thread a OR b]: job1 finishes
[thread a AND b]: exit, all jobs done
[main thread]: join threads, exit
MP6

• We provide you some tools you can use, if you’d like:
  – `queue.c`: A queue data structure
  – `parser.c`: A parser for makefiles
    • `parser_parse_makefile(...)` takes function pointers as arguments that will be called when it reaches a key, dependency, or command.
MP6 Parser Callbacks

parsed_new_key(key=job1)
parsed_dependency(key=job1, dep=job2)
parsed_dependency(key=job1, dep=job3)
parsed_command(key=job1, command=...)
parsed_command(key=job1, command=...)
parsed_new_key(key=job2)
parsed_command(key=job2, command=...)
parsed_new_key(key=job3)
parsed_command(key=job3, command=...)


MP6

• Some useful functions:
  – pthread_create(), pthread_join()
  – sem_init(), sem_wait(), sem_post(), sem_destroy()
  – system()
    • Does fork(), exec(), and wait() for you in one command!

• Remember to check return values! You may find some weird things going on with semaphores if you don’t... Good luck!
• Run a rule only if any dependency has a modification time more recent than the target.

• You can get the modification time of a file using `stat()`
Coding Examples

• This week:
  \[
  \text{ds/ds6/}
  \]