Assignment Review

• Objective
  – fork()
  – exec()
  – wait()

• Two executables
  – Coordinator
  – Checker
Assignment Review

• Checker
  – Two command line arguments
    • Divisor
    • Dividend
  – Checks whether the dividend is evenly divisible by the divisor.
  – Return 1 or 0.

• Implement Checker by itself first

• Then implement Coordinator to manage process execution
Assignment Review

• Coordinator
  – Five command line arguments
    • Divisor
    • 4 dividends
  – 4 cycles
    • fork()
    • exec()
    • wait()
Coordinator Behavior
fork()

- Spawns new child process
- Child process is exact copy of parent
- Returns a different value in parent and child
  - Child’s process id in *parent*
  - Zero in *child*
- Child can get its own process id by using *getpid()*
  - Must include `<unistd.h>`
exec()

• Replaces the contents of a process with a new program.

• execl()
  – More info: type man 2 exec in a shell

• Arguments:
  – Path and executable name
  – Executable name
  – Optional arguments
  – Terminated by NULL
Wait()

• Coordinator should wait until the child completes it’s execution process
• One checker process should be active at a time
• wait()
  – WEXITSTATUS() allows parent to get exit status code of child (see man 2 wait)
Q: What would happen if Coordinator did not call `wait()`?
#CC = clang
CFLAGS += –Wall

all: checker coordinator

checker: checker.c
   $(CC) $(CFLAGS) $< -o$@

coordinator: coordinator.c
   $(CC) $(CFLAGS) $< -o$@

clean:
   rm -f checker coordinator
Requirements

• Code **must** run on lab machines
• Submit all .c and .h files, along with makefile
• Makefile should perform make all and make clean targets
References

• http://www.yolinux.com/TUTORIALS/ForkExecProcesses.html