UTILITIES UNLEASHED

WHAT YOU WILL LEARN

- Fork-Exec
    - Concept of fork and wait
    - How to run program by exec functions
- String Manipulation
    - Find string in string
    - Split strings

FORK & EXEC

- pid_t pid = fork();
- if (pid > 0) {
    - // parent
    - waitpid(pid, &status, 0);
- }
- else if (pid == 0) { // child
    - execvp(command, args);
- }
- else { // fork failed}

EXEC FAMILY

- #include <unistd.h>
- extern char * environ; // get environment variable here
- pass each arguments separately
  - execl(path, arg, …) // execute the file in path
  - execlp(file, arg, …) // look for file if '/' is not contained in file string
  - execle(path, arg, …, envp[]) // execute the file in path + environment setting
- pass an array of string as arguments
  - execv(path, argv[]) // execute the file in path
  - execvp(file, argv[]) // look for file if '/' is not contained in file string
  - execvpe(file, argv[]), envp[]) // environment setting

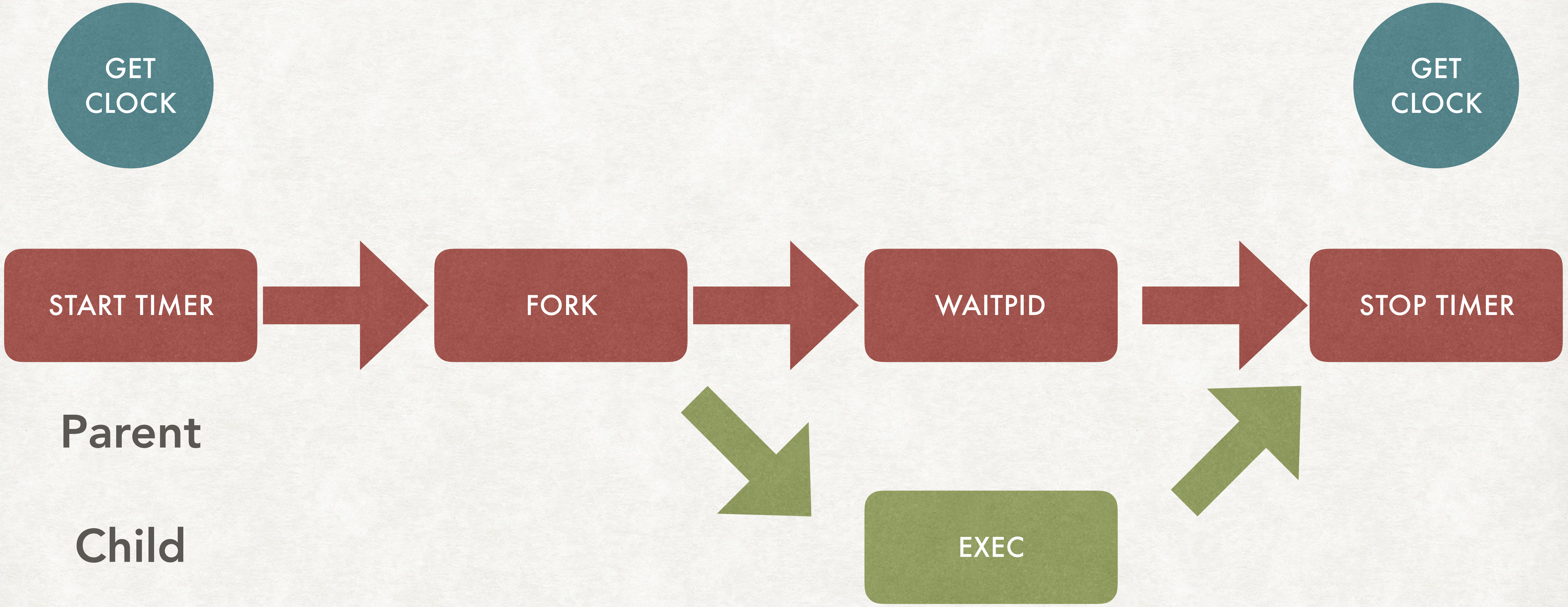
TIME.C

SPEC & THINGS TO KNOW

- ./time <command> <args> …
  - measure the time of running “./sleep 2”
  - should use fork-exec scheme.
  - should take care of programs do not terminate successfully.
  - arguments for the command is not limited two one
    - make -j4 debug
  - use only functions in format.h to print

TIME.C

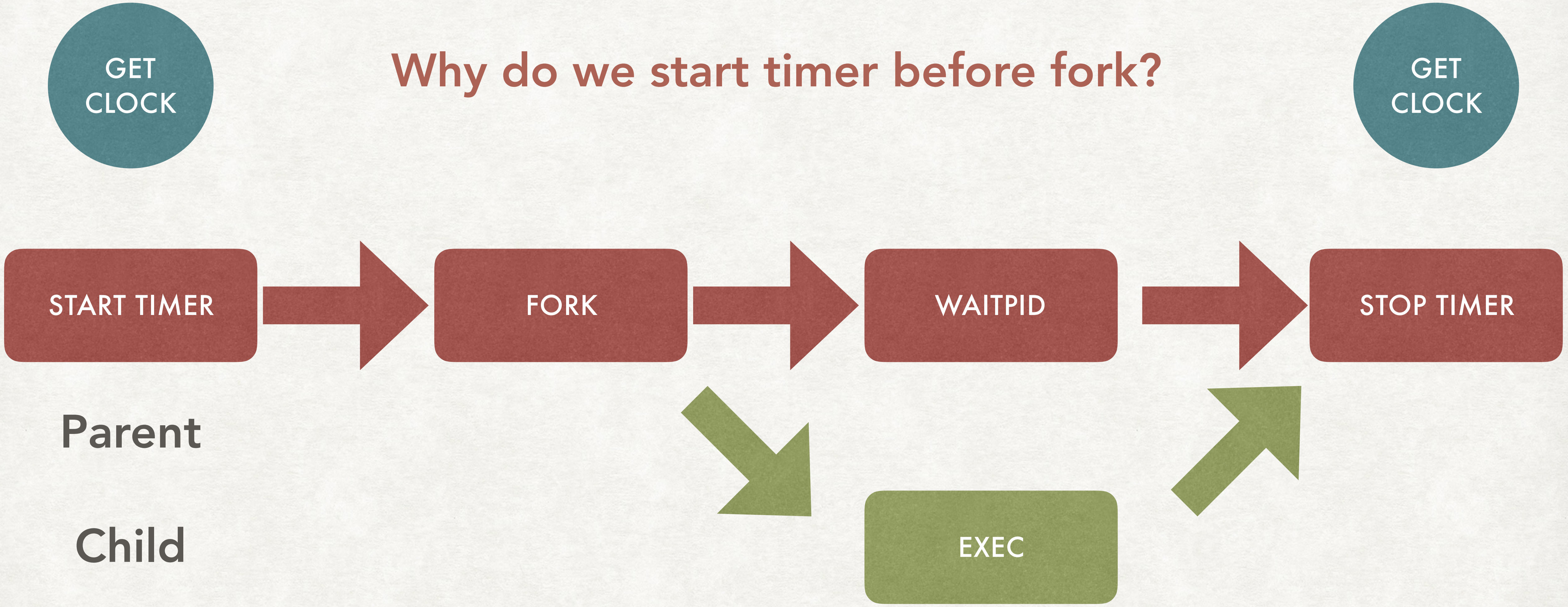
WORKFLOW



TIME.C

WORKFLOW

Why do we start timer before fork?



TIME.C

USEFUL FUNCTIONS

- struct timespec
  - time_t tv_sec; // seconds
  - long tv_nsec; // nanoseconds
  - example tv_sec = 10, tv_nsec = 992300000 -> 10.9923 sec
- int clock_gettime(clockid_t , struct timespec * tm_spec);
  - clockid_t: should use CLOCK_MONOTONIC in this lab
  - get time from tm_spec
  - return 0 when success, -1 otherwise

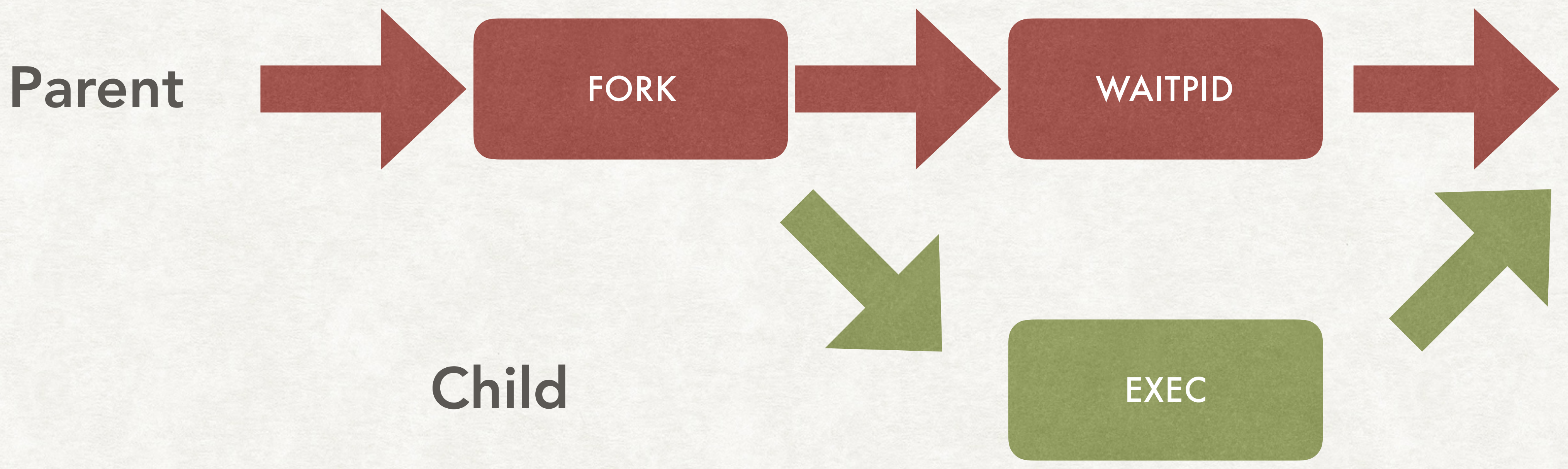
ENV.C

GET & SET ENVIRONMENT VARIABLES

- ./env <var-list> <command-name>
- ./env shows environment variables
- ./env TZ=MST7MDT date
  - execute date under environment TZ=MST7MDT
- use fork-exec
- ./env PATH=%HOME/bin:%PATH make -j4
  - expand variables

ENV.C

WORKFLOW



ENV.C

USEFUL FUNCTIONS

- #include <stdlib.h>
- int setenv(const char* name, const char* value, int overwrite);
    - int flag = setenv(“path”, new_path,1);
- char * getenv(const char *name);
    - char* path = getenv(“path”);

HINTS

DONEC QUIS NUNC

- Usage of argv
  - ./env <some-envirnoment setting> make -j4
    - argv[0] = ./env, argv[1] =< … >, argv[2] = make, argv[3] = -j4
    - execvp(cmd, args) -> execvp(argv[2], argv+2)
- write a split function that can split string based on ‘,’
- write a function that can find all %notation in a string
  - extend that function so that you can replace variables with environment variables
  - use getenv to get environment variables
- be familiar with: return array of strings, clear an array of strings-> camelCasers