

W UNIVERSITY of WASHINGTON L23: Concurrency and Threads CSE333, Summer 2020

Creating and Terminating Threads

- ❖ `int pthread_create(pthread_t* thread, const pthread_attr_t* attr, void* (*start_routine)(void*), void* arg);`
 - Creates a new thread into `*thread`, with attributes `*attr` (`NULL` means default attributes)
 - Returns `0` on success and an error number on error (can check against error constants)
 - The new thread runs `start_routine(arg)`
- ❖ `void pthread_exit(void* retval);`
 - Equivalent of `exit(retval);` for a thread instead of a process
 - The thread will automatically exit once it returns from `start_routine()`

4

4

W UNIVERSITY of WASHINGTON L23: Concurrency and Threads CSE333, Summer 2020


What To Do After Forking Threads?

- ❖ `int pthread_join(pthread_t thread, void** retval);`
 - Waits for the thread specified by `thread` to terminate
 - The thread equivalent of `waitpid()`
 - The exit status of the terminated thread is placed in `**retval`
- ❖ `int pthread_detach(pthread_t thread);`
 - Mark thread specified by `thread` as detached – it will clean up its resources as soon as it terminates

5

5

W UNIVERSITY of WASHINGTON L23: Concurrency and Threads CSE333, Summer 2020

 Poll Everywhere pollev.com/cse33320su

- ❖ Idea: leave a note!
 - Does this fix the problem?

```

if (!note) {
    if (!milk) {
        leave note
        buy milk
        remove note
    }
}

```

A. Yes, problem fixed
 B. No, could end up with no milk
 C. No, could still buy multiple milk
 D. We're lost...

16

16

W UNIVERSITY of WASHINGTON L23: Concurrency and Threads CSE333, Summer 2020

pthread and Locks

- ❖ Another term for a lock is a **mutex** ("mutual exclusion")
 - `pthread.h` defines datatype `pthread_mutex_t`
- ❖ `int pthread_mutex_init(pthread_mutex_t* mutex, const pthread_mutexattr_t* attr);`
 - Initializes a mutex with specified attributes
- ❖ `int pthread_mutex_lock(pthread_mutex_t* mutex);`
 - Acquire the lock – blocks if already locked
- ❖ `int pthread_mutex_unlock(pthread_mutex_t* mutex);`
 - Releases the lock
- ❖ `int pthread_mutex_destroy(pthread_mutex_t* mutex);`
 - "Uninitializes" a mutex – clean up when done

22

22