Pthreads overview and example
Pthreads Programming

PThreads: The POSIX threading interface

- **Portable Operating System Interface for UNIX**
- A standard Interface to OS utilities

System calls have same prototype & semantics on all OSs
POSIX compliant code on Solaris will compile on Linux

Pthreads library contains functions for:

- Creating parallelism
- Synchronizing threads
  - Coordinating their access to shared state

To compile: `gcc myprog.c -lpthread`
Common pthread functions

Creating a thread (starts running start_func w/passed args):

```
int pthread_create(
    pthread_t *thread,
    pthread_attr_t *attr,
    void *(*start_func)(void *),
    void *args);
```

Joining (reaping) a thread (wait for thread to exit):

```
int pthread_join(pthread_t thrd, void **retval);
```

Terminating a thread:

```
void pthread_exit(void *retval)
```

(or just return from thread’s main function)
**void***

int pthread_create(…,  void *args);

**void***: a pointer to any type (a generic pointer)

- all addresses are the same number of bytes
  char *cptr; int *ptr; // store 4 byte addresses

- can pass the address of any type as a void *
  pthread_create( ... , &x); // addr of an int
  pthread_create(..., &ch); //addr of a char

- cannot de-reference a void * pointer
  *args = 6;  // store 6 in 1 byte? 2 bytes? 4 bytes?

- re-cast first before dereference
  *((int *)args) = 6;  // store 6 in 4 bytes
Example

static int x;

int foo(int *p) {
    int y;
    y = 3;
    y = *p;
    *p = 7;
    x += y;
}

If threads i and j both execute function foo code:

Q1: which variables do they each get own copy of? which do they share?
Q2: which stmts can affect values seen by the other thread?
Concurrent Execution

main thread

pthread_create()

pthread_create()

pthread_join()
main thread waits for
thread 1 to terminate

pthread_join() returns

pthread_join() returns

exit()
terminates
main thread and
any peer threads

peer thread 1

printf()
return NULL;

peer thread 2

printf()
return NULL;
(peer thread terminates)
Example: hello.c

• Code on slides is subset of full code
  • Minus error detection and handling for space
  • Minus other non-thread important code

• You can copy and try out full code from here:

```bash
cd ~/cs31/weeklylabs
cp   ~adanner/public/cs31/week12/* .
cd week12
make
./hello 5  # run a few times & try with diff command line arg for num threads
```
hello.c: main function

static unsigned long long count = 0;  // global variable

int main(int argc, char *argv) {
    pthread_t *tids;
    int ntids, i, *tid_args;

    ntids = atoi(argv[1]);
    tids = (pthread_t *)malloc(sizeof(pthread_t)*ntids);
    tid_args = (int *)malloc(sizeof(int)*ntids);

    for (i=0; i < ntids; i++) {
        tid_args[i] = i;
        pthread_create(&tids[i], 0, thread_hello, &tid_args[i]);
    }
    for (i=0; i < ntids; i++) {
        pthread_join(tids[i], 0);
    }
    ...
}

thread_hello:  // each spawned thread’s “main” function
tid_args[i]:   // each spawned thread’s logical tid
tids[i]:       // each spawned thread’s pthread tid
Hello.c: main function:

pthread_create:

```c
err = pthread_create(&tids[i], NULL,
    thread_hello, &tid_args[i])
```

- **function pointer argument** (`thread_hello`)
  - name of the function that the spawned thread will start executing
  - function pointer type:
    ```c
    void *thread_main_func(void *arg);
    ```
  - **args argument** (`tid_args`)
    ```c
    void *: pass a pointer to any type: int, float, struct, array, ...
    ```

pthread_join: like wait in fork-wait

- tid argument: which pthreads thread to wait to exit
hello.c: thread main function

static unsigned long long count = 0; // global variable

void *thread_hello(void *arg) {
    int myid, i;
    myid = *((int *)arg);

    printf("hello I'm thread %d with pthread_id %lu\n", myid, pthread_self());

    for(i = 0; i < 1000000; i++) {
        count += i;
    }

    printf("goodbye I'm thread %d\n", myid);
    return (void *)0; // recast 0 to return type (void *)
}