CS 240
Computer Systems

#8: Limited Direct Execution and Threads

February 10, 2022 · Wade Fagen-Ulmschneider

Example: Joining Threads

07/fifteen-join.c

```c
13 int main(int argc, char *argv[]) {
14   // Create threads:
15   int i;
16   pthread_t tid[num_threads];
17   for (i = 0; i < num_threads; i++) {
18     int *val = malloc(sizeof(int));
19     *val = i;
20     pthread_create(&tid[i], NULL,
                              thread_start, (void *)val);
21   }
22
23   // Joining Threads
24   for (i = 0; i < num_threads; i++) {
25     pthread_join(tid[i], NULL);
26   }
27
28   printf("Done!\n");
29   return 0;
30 }
```

`pthread_join` – In the above program, we use **`pthread_join`**. This call will block the CPU from running the program further until the specified thread has **finished and returned**.

Q1: What happens in this program?

Q2: Does the order vary each time we run it? What is happening?

Q3: What can we say about the relationship between "Done" and "Thread %d running..." lines?

Program Execution: Direct Execution Model

Operating System		User Thread
1. Create entry for process and thread 2. Allocate memory for process and thread 3. Load program into memory 4. Set up stack (argv/argc) 5. Clear registers 6. Call main()	⇒	*(OS has CPU control.)*
(OS does not have CPU control.)		7. Run main() 8. return from main()
9. Free memory for process 10. Remove process from process list	⇐	*(OS has CPU control.)*

What is the problem with this model?

Addition of "Protection Levels":

Limited Direct Execution:

Instead of handing the CPU over to a user thread with full access, a protection mode is set on the CPU that limits the operations a CPU can perform to only operations that do not impact system resources:

Operating System		User Thread
1. Process Init 2. return-from-trap	⇒ Save and Clear Registers Set to "user mode"	*(OS has CPU control.)*
(OS does not have CPU control.)		3. Run main() 4. Makes a system call ...calls **trap-to-OS**
5. Trap Handler ...do syscall work... 6. return-from-trap	⇐ Save+Swap Registers Set to "kernel mode"	*(OS has CPU control.)*
(OS does not have CPU control.)	⇒ Save+Swap Registers Set to "user mode"	...execution continues...

Trapping to the OS: More than Just System Calls

There are several mechanisms to regain CPU control from an application back to the OS:

1. **System Calls**:

2. _____

   What is the purpose of interrupts?

   Are interrupts common?

   Examples:

3. _____

   Examples:

Additional Reading: "Operating Systems: Three Easy Pieces"
Ch. 6: Direct Execution (https://pages.cs.wisc.edu/~remzi/OSTEP/)

Five-State Thread Model

When the operating system has control over the CPU and needs to decide what program to run, it must maintain a model of all threads within the CPU.

We commonly refer to the "state" of a thread as part of the five-state model:

Counting with Threads

Here's a new program using multiple threads, which we will compile as the executable **count** (`gcc count.c -o count`):

08/count.c

```
 5 int ct = 0;
 6
 7 void *thread_start(void *ptr) {
 8   int countTo = *((int *)ptr);
 9
10   int i;
11   for (i = 0; i < countTo; i++) {
12     ct = ct + 1;
13   }
14
15   return NULL;
16 }
17
18 int main(int argc, char *argv[]) {
 …   /* [...check argv size...] */
24
25   const int countTo = atoi(argv[1]);
 …   /* [...error checking...] */
28   const int thread_ct = atoi(argv[2]);
 …   /* [...error checking...] */
30
31   // Create threads:
32   int i;
33   pthread_t tid[thread_ct];
34   for (i = 0; i < thread_ct; i++) {
35     pthread_create(&tid[i], NULL,
          thread_start, (void *)&countTo);
36   }
37
38   // Join threads:
39   for (i = 0; i < thread_ct; i++) {
40     pthread_join(tid[i], NULL);
41   }
42
43   // Display result:
44   printf("Final Result: %d\n", ct);
45   return 0;
46 }
```

Q1: What do we expect when we run this program?

Q2: What is the output of running:
`./count 100 2`

Q3: What is the output of running:
`./count 100 16`

Q4: What is the output of running::
`./count 10000000 2`

Q5: What is the output of running::
`./count 10000000 16`

Q6: What is going on???