2

Putting it All Together: An Example Session with OSP 2

2.1 Chapter Objective

Your instructor has assigned you the Threads project to implement; see Chapter 4. You are new to OSP 2. What do you do? In this chapter, we present an example session with OSP 2 that is intended to give you the guidance and confidence you need to successfully complete your assignment.

2.2 Overview of Thread Management in OSP 2

The Threads project, as the name implies, deals with thread management and scheduling, where threads are the executable and dispatchable units in OSP 2. Our example will focus on thread management, in particular, the resumption of a thread from a waiting state. This activity is the responsibility of the method do_resume(), one of the methods you are to implement as part of your implementation of the class ThreadCB.

Thread management involves the notions of thread creation, destruction, suspension, resumption and dispatching; maintaining thread status; and moving threads between different (ready and waiting) queues. Underlying all of this is the notion of a thread state, which can be one of ThreadReady, ThreadWaiting, ThreadKill, etc.
An OSP thread assumes the ThreadWaiting state when it enters the pagefault handler or when it executes a blocking system call (e.g., `write()`). The ThreadWaiting state is also known as the “level-0 waiting state”. While in this state, a thread can again enter the pagefault handler or execute a blocking system call, causing it to enter the level-1 waiting state, represented by the constant ThreadWaiting+1. This process can continue indefinitely, leading to arbitrarily nested depths of waiting.

When a thread completes the execution of the pagefault handler or blocking system call, it should be moved up to the next highest waiting level by decrementing its waiting status; in the case of level 0 (ThreadWaiting, it should transit to the ThreadReady state.

### 2.3 The Student Method `do_resume()`

As mentioned previously, we will focus our attention during this example session on the method `do_resume()` of class ThreadCB. Its code is given in Figure 2.1. Notice the use of the MyOut utility to insert student output in the file `OSP.log`. For example, the statement

```java
MyOut.print(this, "Resuming " + this);
```

will result in output such as

```
Mod: 63 [Threads.ThreadCB]
    Resuming Thread(0:1/W2)
```

appearing in the log file, indicating that at simulation time 63, thread 0 of task 1 is at waiting-level 2 (W2). The tag “Mod:” identifies this output as being from a student module, making it easy for you to distinguish your output from OSP’s in the log file.

`do_resume()` is one of the simplest methods in OSP. All it needs to do is decrement the thread’s waiting-level, place it on the ready queue if its new status is ThreadReady, and call `dispatch()` so that some thread can be dispatched onto the CPU for execution.

Assuming that you have completed your design and coding of the Threads project, let us proceed in a step-by-step fashion with the example session.