cs452 – Spring 2016
Kernel 3

Bill Cowan
University of Waterloo

I. Introduction

The third part of your kernel adds interrupts generated by a counter-timer. You use this capability to create a clock server. To do so you must have working
1. an implementation of AwaitEvent,
2. an implementation of an idle task, or equivalent,
3. implementations of a clock server and a clock notifier, and
4. implementations of Delay, Time and DelayUntil as wrappers for Send to the clock server.

Tasks that want to obtain service from the clock server must do so by obtaining its tid from the name server.
Comment. Most students use a single kernel entry point for hardware and software interrupts. The set of things needed to handle software interrupts is a subset of the set needed for hardware interrupts so a correct implementation of hardware interrupts works fine for software ones. This shrinks the code size at the cost of executing some extra instructions.

II. Description

II.1. Kernel

To accomplish this part of the kernel you must have the following kernel primitive operating:
- `int AwaitEvent( int eventType ).`

See the kernel description and the lecture notes to read the details of its operation.
In addition you must program a first user task, which
- creates the name server,
- creates the clock server, and
- creates the clock server's clients.

II.2. The Clock Server and Notifier

The clock server should be implemented as described in the kernel documentation. The clock notifier should wait on events from a timer, and Send notifications that the timer ticked to the clock server.

II.3. Client tasks.

A single type of client task tests your kernel and clock server. It is created by the first user task, and immediately sends to its parent, the first user task, requesting a delay time, t, and a number, n, of delays. It then uses `WhoIs` to discover the tid of the clock server.

It then delays n times, each time for the time interval, t. After each delay it prints its tid, its delay interval, and the number of delays currently completed on the terminal connected to the ARM box.

II.4. First user task

The first user task creates the clock server, and four client tasks. It then executes Receive four times, and Replies to each client task in turn. The following table shows the parameters to be given to the clients.

Priority (smaller is higher)	Delay Times (ticks)	Number of Delays
3	10	20
4	23	9
5	33	6
6	71	3

II.5. Idle task

During the time that all client tasks are delayed, an idle task should run. You can use the idle task to measure and report on operating characteristics of your system. One operating characteristic you must produce for this assignment is the fraction of execution time used by the idle task. This must be shown on the terminal display.

We will expect you to maintain this performance indicator throughout the remainder of the term. You will find it an invaluable diagnostic of performance problems.

III. Hand in

Hand in the following, nicely formatted and printed.

1. A description of how to access, make and operate your program.

2. A description of the structure of your kernel so far. We will judge your kernel primarily on the basis of this description. Describe which algorithms and data structures you used and why you chose them.
3. The location of all source code you created for the assignment and either MD5 hashes of each file or an SHA1 hash of your repository. The code must remain unmodified after submission until the assignments are returned.
4. Output produced by your client tasks and an explanation of why it occurs in the order it does.