Functional programming in Erlang

Grupo de Sistemas Distribuídos
Universidade do Minho

Objectives

Getting familiar with Erlang compiler and runtime environment. Exploring the Erlang functional core, implementing and using purely functional abstract data types.

Mecanisms

Modules, functions, lists and tuples.

Tasks

1. Write a module `myqueue` that implements a queue abstract data type, that allows item insertion and removal with FIFO semantics. The functions to be made available are:

   ```
   create() -> Queue
   enqueue(Queue, Item) -> Queue
   dequeue(Queue) -> empty | {Queue, Item}
   ```

   These operations should allow creating an empty queue, inserting an element, and removing an element (if possible), respectively. Test the queue module by writing another module which makes use of it.

2. Write a module `priorityqueue`, for an abstract data type which generalizes the queue above, with the notion of priority. A dequeue should now remove the oldest item having the smallest priority class. The functions to be made available are:

   ```
   create() -> PriQueue
   enqueue(PriQueue, Item, Priority) -> PriQueue
   dequeue(PriQueue) -> empty | {PriQueue, Item}
   ```

   Implement two versions of `priorityqueue`, each making use of the `myqueue` module:

   (a) assuming only a few priorities will be used, use a list of queues for the state;

   (b) use the library module `gb_trees` to map priorities to queues.