# Synchronization Variables

CML's `SyncVar` structure provides two kinds of synchronous variables: *incremental* variables, or *I-vars*, and mutable variables, or *M-vars*. These variables have two states: empty and full. I-vars are write-once variables, whereas M-vars may be written, emptied, written again, etc.

I-vars and M-vars could be implemented in terms of channels, but they are actually implemented in a more efficient way. It is more efficient to use I-vars for communicating replies, than to use reply channels.

# Synchronization Variables (Cont.)

Here is part of the signature of the `SyncVar` structure:

```
exception Put  (* raised when writing to non-empty var *)

type 'a ivar  (* pointer to I-var *)
val iVar    : unit -> 'a ivar
val iPut    : 'a ivar * 'a -> unit
val iGet    : 'a ivar -> 'a
val iGetEvt : 'a ivar -> 'a CML.event

type 'a mvar  (* pointer to M-var *)
val mVar     : unit -> 'a mvar
val mPut     : 'a mvar * 'a -> unit
val mTake    : 'a mvar -> 'a              (* empties *)
val mTakeEvt : 'a mvar -> 'a CML.event  (* empties on sync *)
val mGet     : 'a mvar -> 'a
val mGetEvt  : 'a mvar -> 'a CML.event
```

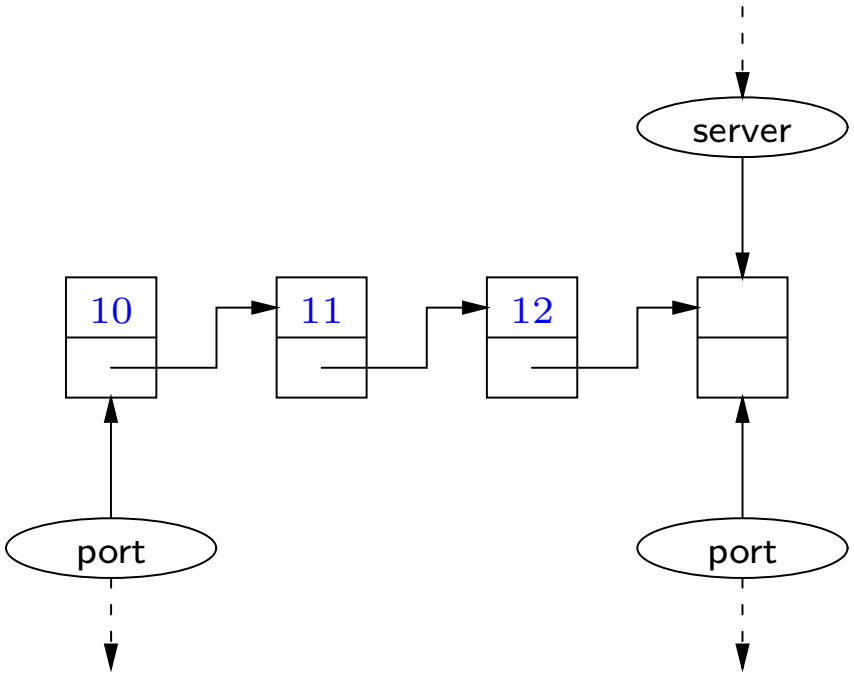
# Multicasting

A multicast channel is a way of sending messages to an arbitrary number of listeners via ports onto the multicast channel. When a message is sent to the multicast channel, it is available via all ports onto the multicast channel that existed when the message was sent.

Multicast channels are provided by CML's `Multicast` structure. But we will re-implement part of this structure, as an example.

# Multicasting (Cont.)

A multicast channel consists of a stream of messages made out of I-vars, a server thread (which takes in messages to be multicasted, plus requests to create new ports), plus some number of port threads.




The server thread keeps a pointer to the unfilled I-var at the end of the stream. And each port thread has a pointer to the I-var whose value will next be supplied to threads receiving on that port.