Architecture for synchronizing concurrent processes that have a shared memory space and a modified termination behavior.

The modified behaviour is that a process may need wait for a set of other processes (non-child processes) before terminating.

Let consumers be the set processes that wait for all producers to exit.

If producing and consuming are considered as two event.

Following flow chart describes how the execution occurs if above two sets of processes are considered.

Time

The **root process** , the **producer(s)** and the **consumer(s)** all reside in the user space and it should be noted that processes cannot read or write each others memory.

STAGE I:

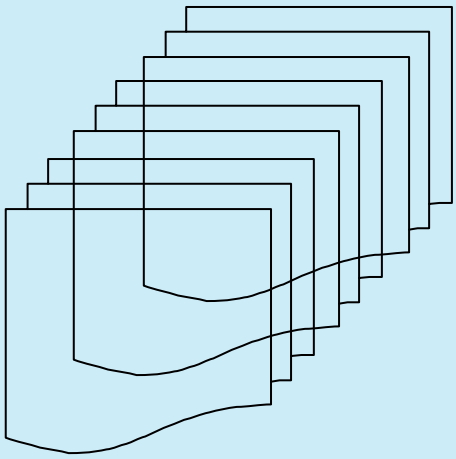
Initialization

At this stage all processes: the root process, event-drivers and event-consumers are generated.

STAGE II:

Address Mapping

Processes residing in memory are need to be mapped to a shared memory range as direct R/W is not allowed between processes.



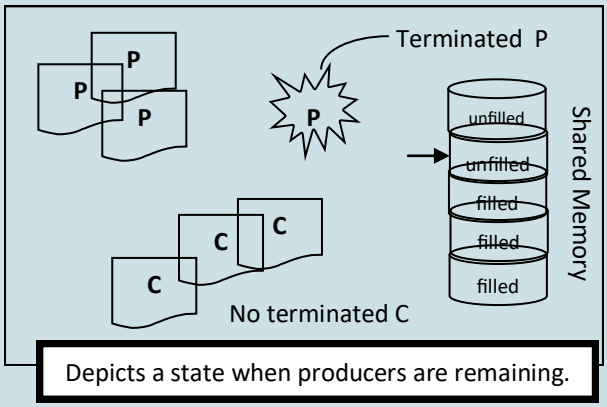
:represents a process

STAGE III/IV/V:

Stage III: produced value is assigned to a shared location. (if available / else wait)

Stage IV: produced value is consumed. (if any / else wait)

Stage V: consumers consume BUT they can not exit unless and until all producers exit (NOTE: a consumer therefore can consume more than one produced value and even none).



Depicts a state when producers are remaining.

- There is synchronization between the three stages.
- For consumers there are two exit section one is to terminate and the other is to loop.
- The code implementation is mainly about the above two point.