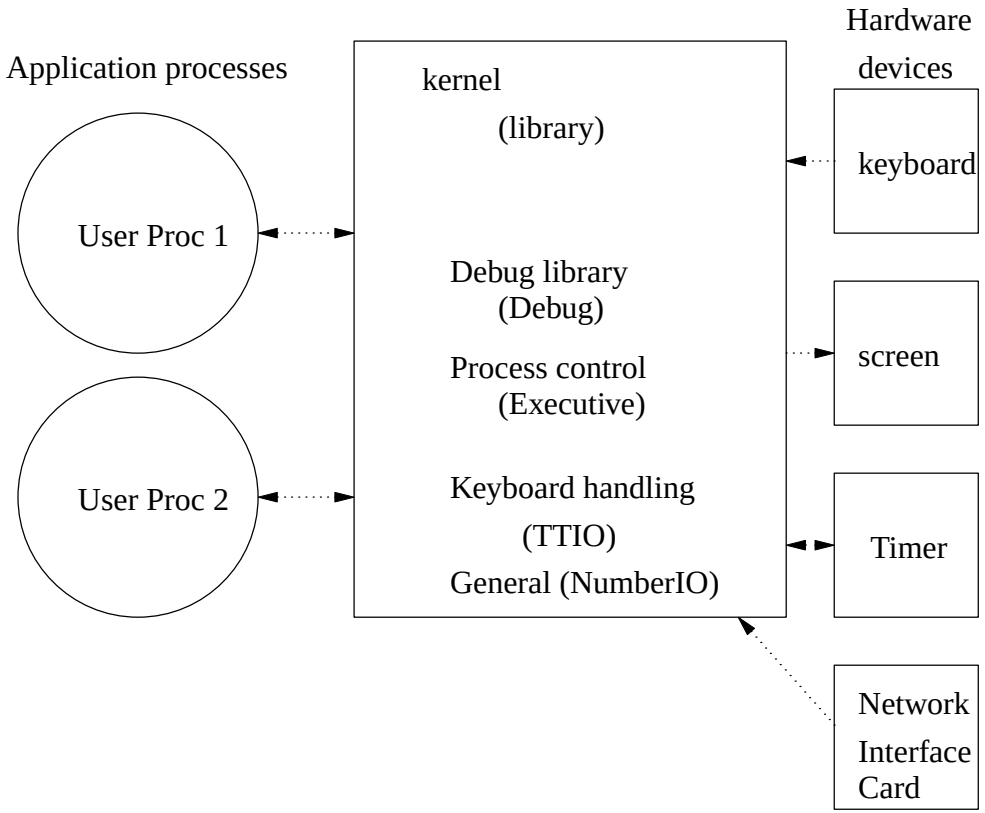
The Microkernel Executive



The Microkernel Executive

- coordinates concurrent process activity within the
  - creation of processes
  - coordination of interrupts
  - scheduling of processes
  - provides mutual exclusion primitives
    - Wait, Signal - Semaphores in this Executive

Functions provided by the Executive

```
Descriptor *
   Executive_InitProcess (void (*p)(void),
                          int StackSize,
                          char *Name,
                          int NameLength);
```

- initializes a process which is held in the suspended state
- when the process is resumed it will start executing procedure, p.
- the process has a maximum stack size of, StackSize, bytes and its textual name is, Name. The StackSize should be at least 5000 bytes.

Functions provided by the Executive

- ```
  void *Executive_Resume (Descriptor *d);
  ```
- resumes a suspended process
- if all is successful then the process, p, is returned
- if it fails then NIL is returned
  ```

Functions provided by the Executive

- ```
  void Executive_Suspend (void);
  ```
- suspend the calling process
- the process can only continue running if another process Resumes it.
  ```

Functions provided by the Executive

```
Semaphore *
   Executive_InitSemaphore (int v,
                            char *Name,
                            int NameLength);
```

- creates a semaphore whose initial value is, v, and whose name is, Name

Functions provided by the Executive

- ```
  void Executive_Wait (Semaphore *s);
  ```

- performs dijkstra's P operation on a semaphore

- a process which calls this procedure will wait until the value of the semaphore is > 0 and then it will decrement this value
  ```

Functions provided by the Executive

- ```
  void Executive_Signal (Semaphore *s);
  ```

- performs dijkstra’s V operation on a semaphore

- a process which calls the procedure will increment the semaphores value
  ```

Functions provided by the Executive

- ```
  EXTERN void Executive_WaitForIO (int VectorNo);
  ```

- waits for an interrupt to occur on vector, VectorNo
  ```

Functions provided by the Executive

- ```
  void Executive_Ps (void);
  ```

- displays a process list together with process status
  ```

Functions provided by the Executive

- ```
  Descriptor *Executive_GetCurrentProcess (void);
  ```

- returns the descriptor of the current running process
  ```

Functions provided by the Executive

- `void Executive_RotateRunQueue (void);`

- rotates the process run queue
  - it does not call the scheduler

Functions provided by the Executive

- ```
  void Executive_ProcessName (Descriptor *d);
  ```

- displays the name of process, d, through DebugString
  ```

Functions provided by the Executive

- ```
  void Executive_DebugProcess (Descriptor *d);
  ```

- gdb debug handle to enable users to debug deadlocked semaphore processes
  ```