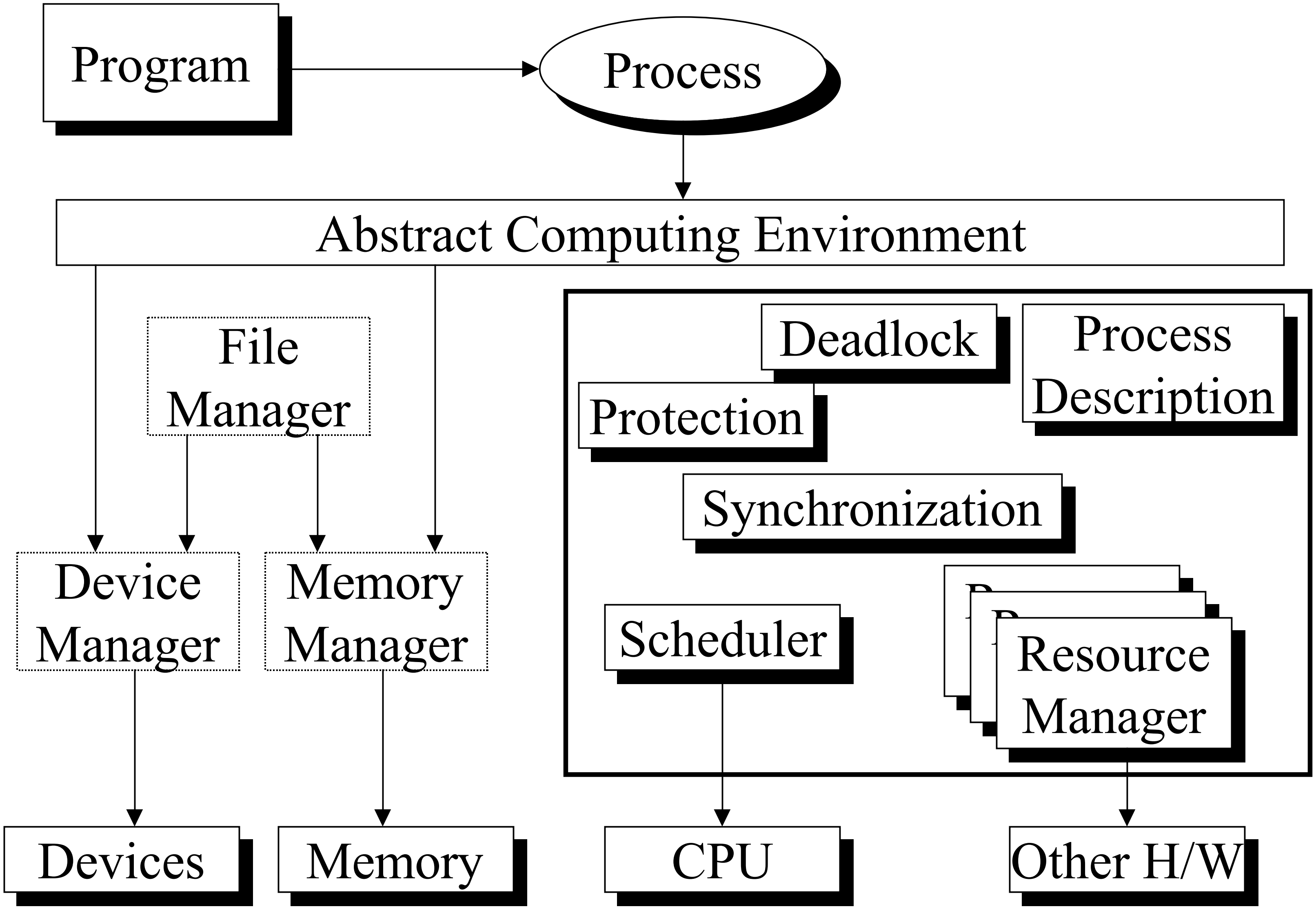
Process Management

Process Manager Overview



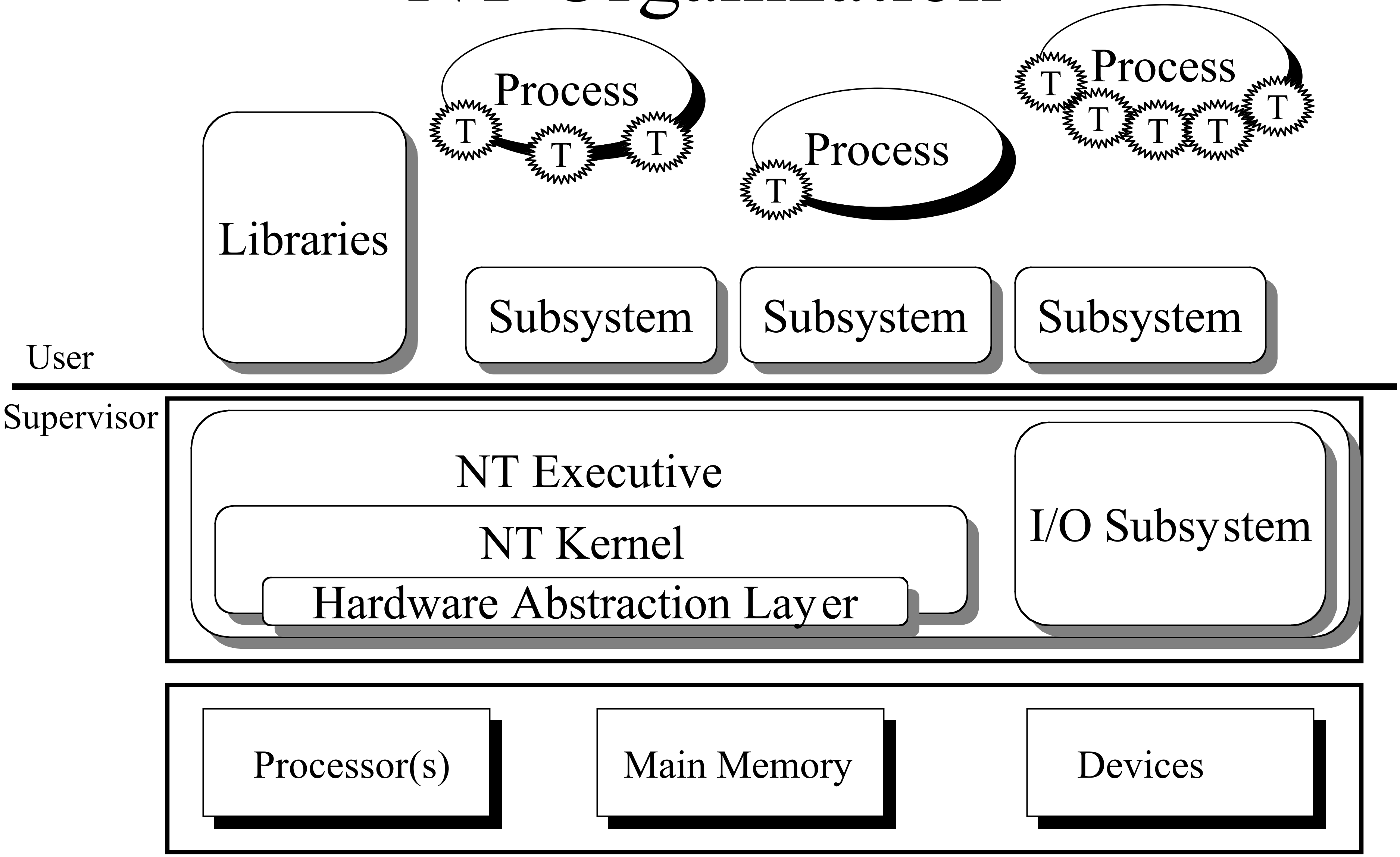
Process Management

- Representing the essential characteristics of a process -- the process descriptor
- What “things” can be referenced -- the address space
- Allocating resources
- Creating/destroying processes
- Scheduling the CPU (Chapter 7)
- Synchronization mechanisms (Chapters 8-9)
- Deadlock (Chapter 10)

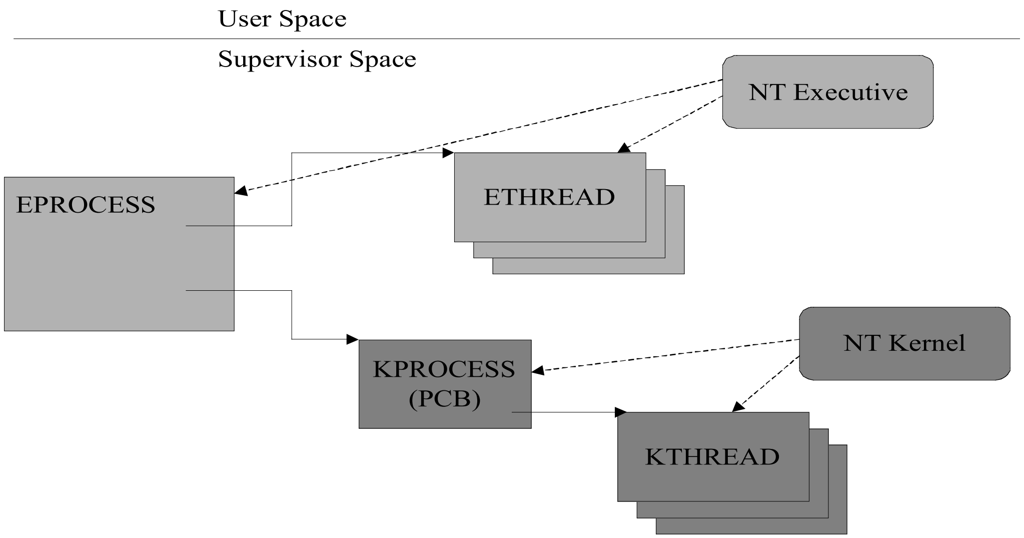
Process Descriptor

- OS creates process abstraction
- Descriptor is data structure for the process
  - Register values
  - Logical state
  - Type & location of resources it holds
  - List of resources it needs
  - Security keys
  - etc. (see Table 6.1 and the source code of your favorite OS)

NT Organization



NT Process Descriptor



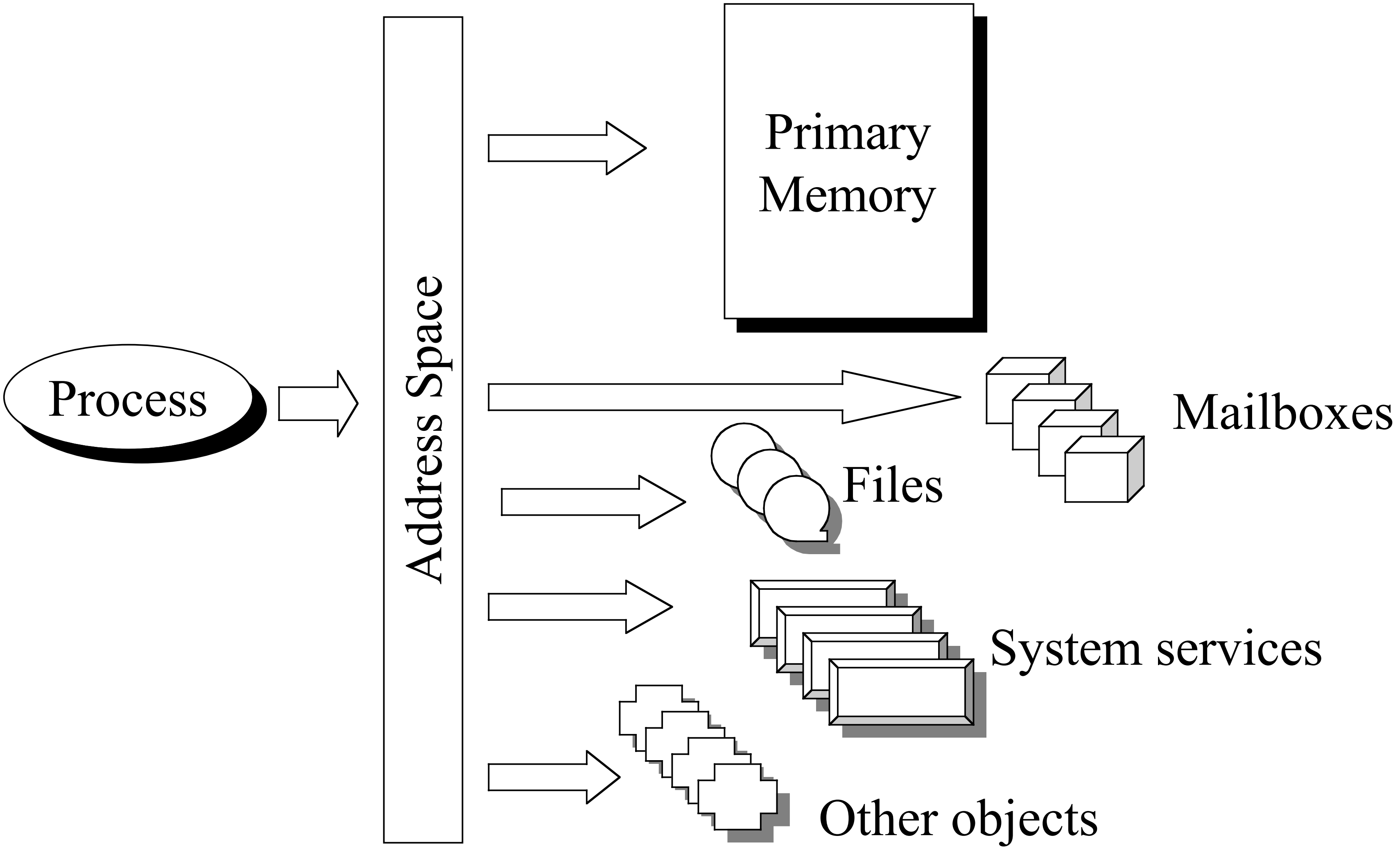
NT Process Descriptor

- Kernel process object including:
  - Pointer to the page directory
  - Kernel & user time
  - Process base priority
  - Process state
  - List of the Kernel thread descriptors that are using this process

NT Process Descriptor (cont)

- Parent identification
- Exit status
- Creation and termination times.
- Memory status
- Security information
- executable image
- Process priority class used by the thread scheduler.
- A list of handles used by this process
- A pointer to Win32-specific information

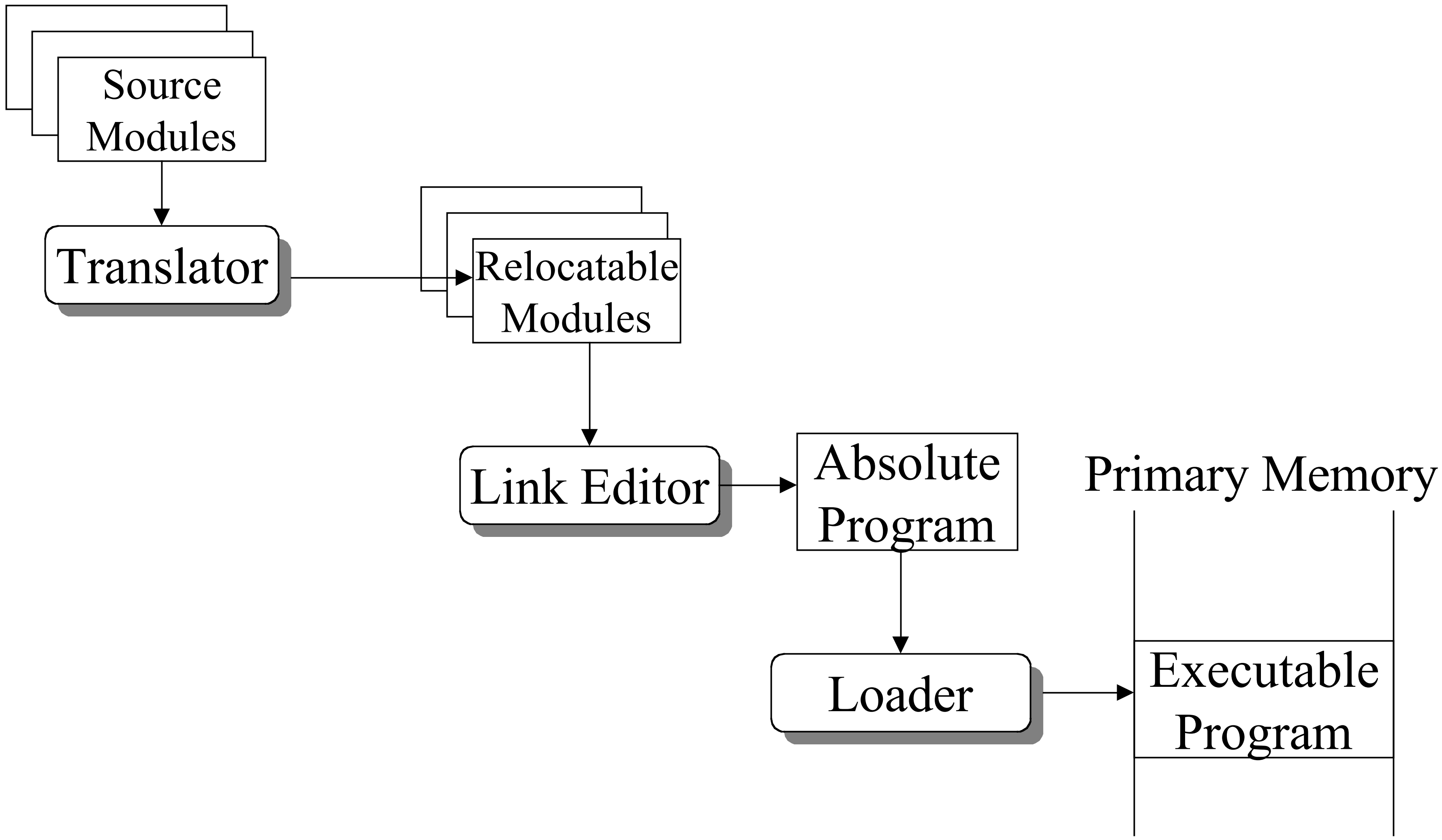
Address Space



Defining the Address Space

- Some parts are built into the environment
  - Files
  - System services
- Some parts are imported at runtime
  - Mailboxes
  - Network connections
- Memory address block is created at compile time

The Compile Time Component



Source Modules

Translator

Relocatable Modules

Link Editor

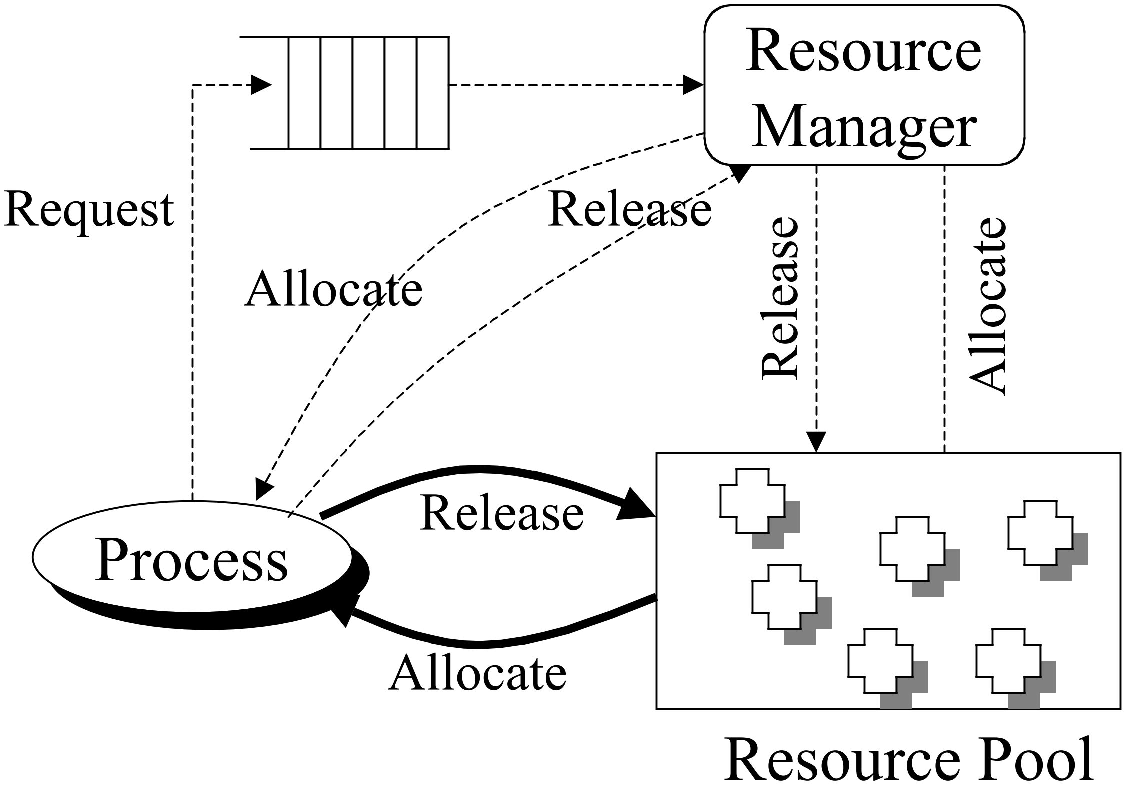
Absolute Program

Primary Memory

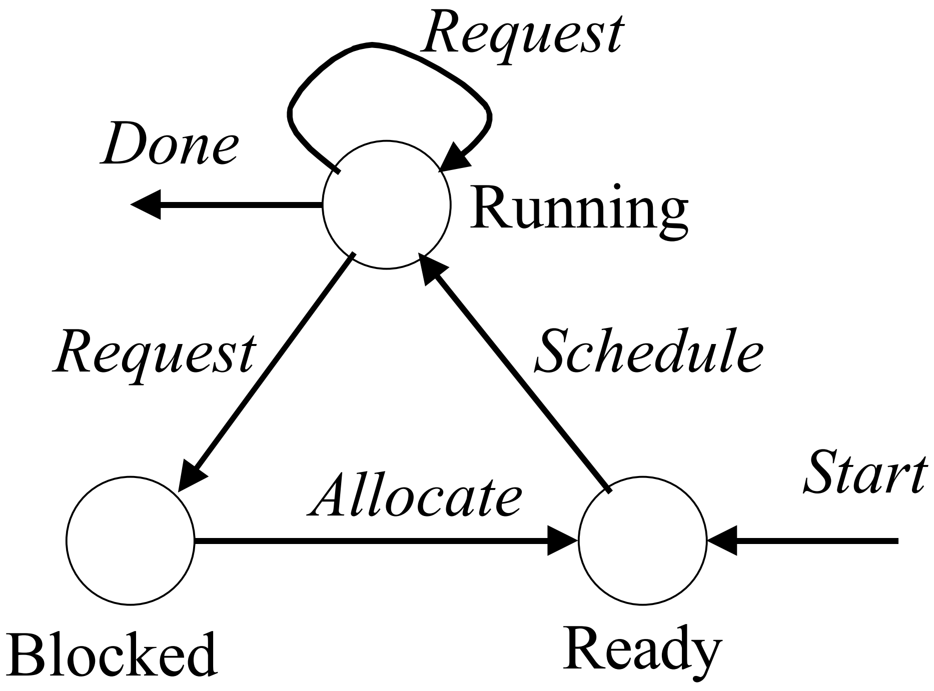
Loader

Executable Program

Resource Manager



Process State (Version 1)



Creating a Process

- Have seen UNIX `fork` & `exec`
- Derived from `FORK`, `JOIN`, & `QUIT`

`FORK(label)`: Create another process in the same address space beginning execution at instruction `label`

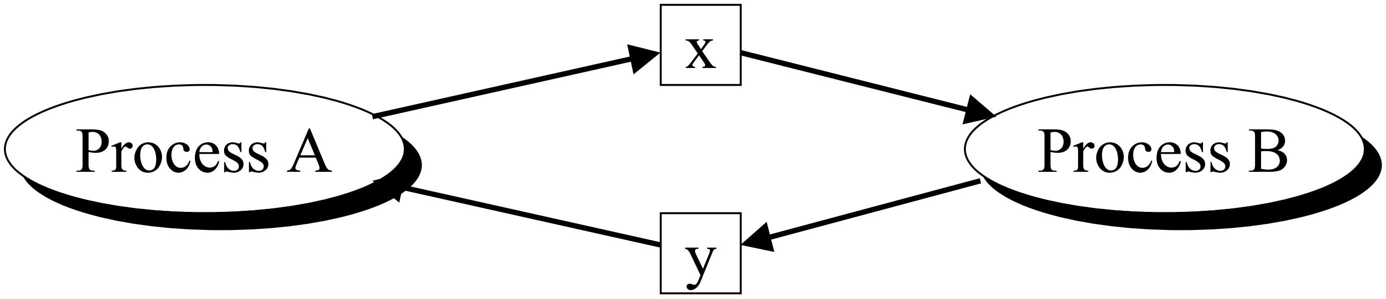
`QUIT()`: Terminate the process.

```
JOIN(count):
    disableInterrupts();
    count--;
    if(count > 0) QUIT();
    enableInterrupts();
```

Example

```
procA() {
  while(TRUE) {
    <compute section A1>;
    update(x);
    <compute section A2>;
    retrieve(y);
  }
}
```

```
procB() {
  while(TRUE) {
    retrieve(x);
    <compute section B1>;
    update(y);
    <compute section B2>;
  }
}
```



Example (cont)

```
L0:  count = 2;
     <compute section A1>;
     update(x);
     FORK(L2);
     <compute section A2>;
L1:  JOIN(count);
     retrieve(y);
     goto L0;
L2:  retrieve(x);
     <compute section B1>;
     update(y);
     FORK(L3);
     goto L1;
L3:  <compute section B2>
     QUIT();
```

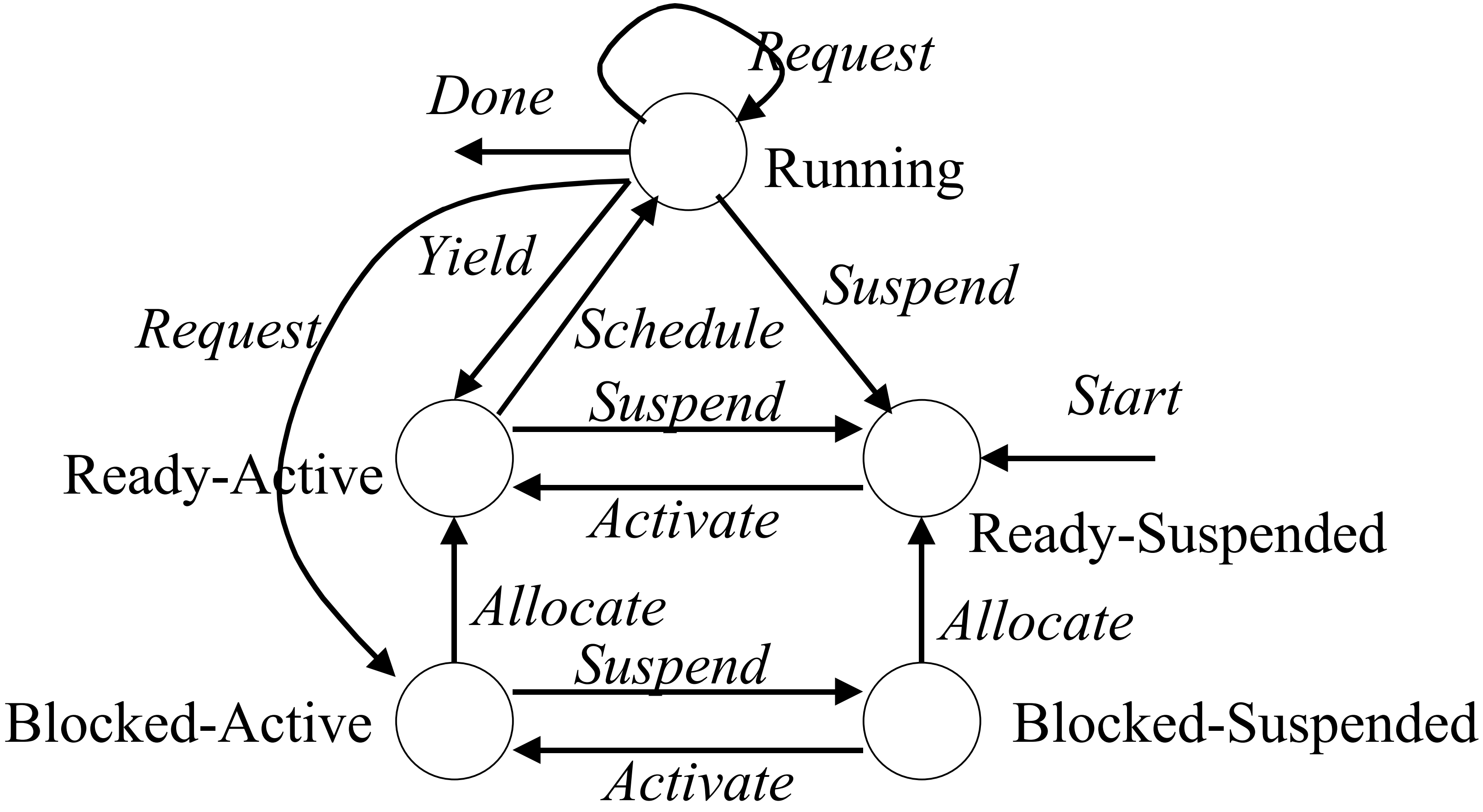
Example (cont)

```
L0:  count = 2;
     <compute section A1>;
     update(x);
     FORK(L2);
     <compute section A2>;
L1:  JOIN(count);
     retrieve(y);
     goto L0;
L2:  retrieve(x);
     <compute section B1>;
     update(y);
     FORK(L3);
     goto L1;
L3:  <compute section B2>
     QUIT();
```

```
L0:  count = 2;
     <compute section A1>;
     update(x);
     FORK(L2);
     retrieve(y);
     <compute section B1>
     update(y>;
     FORK(L3)
L1:  JOIN(count);
     retrieve(y);
     goto L0;
L2:  <compute section A2>;
     goto L1;
L3:  <compute section B2>
     QUIT();
```

Process Hierarchies

- Parent-child relationship may be significant: parent controls children's execution


Request
Done
Running
Yield
Request
Schedule
Suspend
Suspend
Start
Ready-Active
Activate
Ready-Suspended
Allocate
Allocate
Suspend
Blocked-Active
Blocked-Suspended
Activate