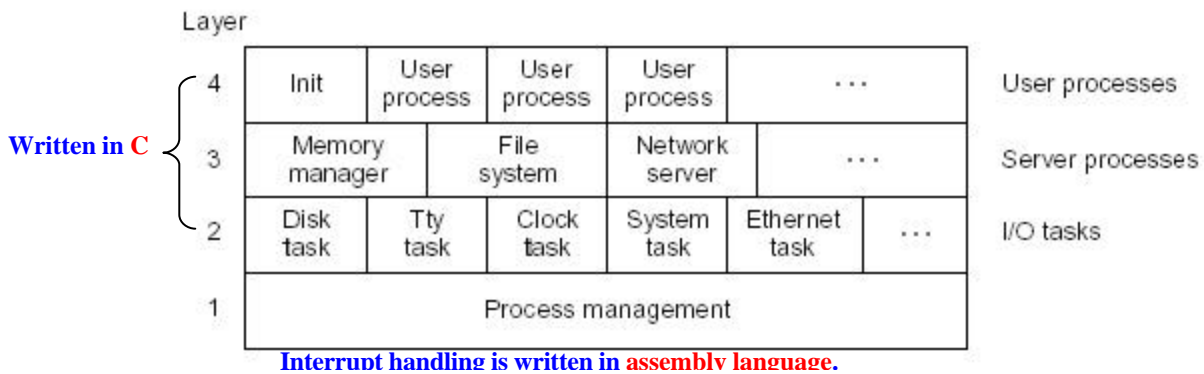


# Overview of Processes in MINIX:

MINIX is structured in four layers:



- The bottom layer **catches** all **interrupts** and traps, **does scheduling**, and provide higher layers with a model of independent sequential processes that **communicate** using **messages**. For the **message passing**, this layer **locates** “send” and “receive” buffers in physical memory, and **copies bytes from sender to receiver**.
- Layer 2 contains the **I/O processes**. It controls **devices** including **disks, printers, terminals, network interfaces, and clocks**. The name of “**task**” means **device drivers**. (See Chapter three for detail) The system task does copying between different memory regions. This layer has less privilege than the bottom layer.
- The first and second layers are glued together and called "**Directory Kernel**" or "**Tasks**".
- Layer 3 contains processes that provide useful services to the user processes. The **memory manager** (MM) carries out all the MINIX system calls that involve memory management, such as **FORK, EXEC, and BRK**. The **file system** (FS) carries out all the file system calls, such as **READ, MOUNT, and CHDIR**. The system call interpretation is in this layer.
- The fourth layer has user processes. e.g. vi editor, Make program, Compiler... etc.
- All the user processes in the whole system are part of a **single tree** with **init** at the **root**. The **shells** are the children of **init**, the **user processes** are the grandchildren of **init**, and the user processes in the system are part of a single tree.
- Once the loading operation is complete the kernel starts running. Then, the memory manager, the file system, and other servers in layer 3 run and initialize themselves. After that, they will be blocked and wait for something to do. At this time, **init** starts running.

## Interprocess Communication in MINIX:

**send**(dest, &message);

to send a message to process, dest.

**receive**(source, &message);

to receive a message from process source (or ANY)

**send\_rec**(src\_dst, &message);

to send a message and wait for a reply from the same process.

MINIX uses the **rendezvous method**: no receiver blocks “send”.

Each process or task can send and receive messages from processes and tasks **in its own layer**, and from those in the layer **directly below it**. User processes may not communicate directly with the I/O tasks

## MINIX Scheduling:

Each time a process is interrupted from a I/O device or from the clock.

The priority: **task --> server --> applications** (user process)

Is any body in **Tasks**?

If yes, run it.

If no, check any body in **Servers**.

If yes, run server.

If no, run **user process**.

Within the task and sever levels processes run until they block, never preempted by the clock, but user processes are scheduled using **round robin**.