A System Architecture, for an Insignificant OS

**Introductions**

The objective of this document is to establish a set of standards that TXP-VOS will follow for the next decade, assuming that it ever sees the light of day. These guidelines are intended to increase the stability, reliability, intuitiveness, and simplicity of the final product. To achieve this, certain measures have been taken in the design of this x86-compatible OS.

**Drivers and Kernel Modules**

While devices like the keyboard and monitor are necessary to even interact with the OS at all, other hardware like mice, NICs, and sound cards are not always necessary. The OS is intended to come with generic, vendor-neutral drivers for a minimal set of hardware out-of-the-box, to get end users up and running as quickly as possible. These drivers will have limited functionality. But they are only meant to allow the user a chance to install vendor-specific drivers for their devices. While they could be used in long-term, doing so is not recommended. This would be akin to using Windows 10 with a GeForce GTX 1080 Ti, and having the OS use the Microsoft's Basic Display Adapter instead of nVIDIA's latest GeForce drivers. Unlike Windows 9X, these included drivers are not real mode drivers. All drivers are run outside of KernelSpace, in either SystemSpace or UserSpace. It is intended that drivers, and most kernel modules, be run in SystemSpace.

**The Basics of INTerrupts**

All Interrupts would go through IOMan (via the PIC), and come in the following variants:

- **Hardware**
- **Software** (typically use the INT instruction):
  - Programmed INT's
  - Traps/Exceptions

Of these categories, Traps/Exceptions tend to be non-maskable. The other categories vary more in this regard. However, the flow of execution always travels in one direction – from a context of similar or reduced privileges to an escalated context with more privileges. Then, it falls back to its original context when the escalation is no longer necessary. Interrupts cannot flow in the opposite direction, and have no practical reason to do so. This form of propagation will come up again later.

**The SystemSpace**

This new space is meant for all modules that are not part of SysMan, or the base kernel. This includes Subsystems like IOMan and DevMan. It also includes 3rd party GPU drivers, audio card drivers, printer/scanner software, and other packages from external vendors. Software components in this space will still receive access to the hardware, through the HAL – which is provided and managed by DevMan. In short, SystemSpace can make the use of Kernel Modules safer and more secure, due to its implementation of reduced privileges when compared to traditional KernelSpace.

SystemSpace can also be used to possibly implement safer in-memory System Calls, regardless of the x86 processor in question. System Calls happen when a process requests a service or access to a resource, from the kernel. This request typically involves temporarily handing over the flow of execution from a user process to the underlying OS. This OS would implement two methods of handling System calls:

---

**Commented [TP1]:** Taken from here: https://softwareengineering.stackexchange.com/a/158064
The definitions given most-closely align with the intended characteristics.

**Commented [TP2]:** Based on information seen here: https://www.techrepublic.com/article/get-it-done-troubleshooting-real-mode-and-protected-mode-drivers/

**Commented [TP3]:** We will use this for the System Call categories: https://www.guru99.com/system-call-operating-system.html#4
A System Architecture, for an Insignificant OS

- Via Interrupts
- Via Arguments, in Memory

All System Calls would go through IOMan, since the requests are communications that go between OS subsystems in this architecture. How they are handled specifically would differ depending on the type. Ones that have to do with Process Control and Information Maintenance would be handled via Interrupts. Those that involve File Management, Device Management, and Inter-Process Communication would be handled in-memory. System Calls made in the SystemSpace are intended to have restrictions that could make them safer to use.

**The System Hierarchy**

This new system hierarchy shows the intended separation of roles that would be present in the OS, and gives minor examples of the elements they are responsible for. From here on, the Man suffix is used as shorthand for the word Manager, and RT is shorthand for RunTime.

<table>
<thead>
<tr>
<th>SysMan (System Manager - a three-part subsystem that lives in KernelSpace)</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Super (Supervisor - PID 0. Process supervision and Watchdog)</td>
</tr>
<tr>
<td>• Synchronos (Time and Sync - PID 1. Timekeeping, Synchronisation, Scheduling, etc.)</td>
</tr>
<tr>
<td>• InitMan (Initialisation - PID 2. Initialises the remaining subsystems)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IOMan (Input/Output; runs in SystemSpace, PID 3. Communications/requests between subsystems)</th>
</tr>
</thead>
<tbody>
<tr>
<td>• MemMan (Memory Manager - paging, protections, garbage collection, segmentation, etc.)</td>
</tr>
<tr>
<td>o RTMan (RunTime Manager – processes, threads, daemons, services etc.)</td>
</tr>
<tr>
<td>• CRTMan (OS-Level Containerisation that isolates all external processes by default)</td>
</tr>
<tr>
<td>• DevMan (Device Manager - HAL, drivers, DMA, MMIO/PMIO, device Polling/INT's, etc.)</td>
</tr>
<tr>
<td>• UIMan (User Experience Manager; runs in UserSpace, SHell, GUI, UserAccountTemplates, etc.)</td>
</tr>
<tr>
<td>o UsrMan (User/Session Manager – for data that is user-specific and persistent)</td>
</tr>
</tbody>
</table>

These subsystems are meant to be resilient, modular, small (in footprint), and recoverable. If one goes down, the entire OS doesn't have to go down with it. That subsystem, that module, can be safely restarted while the system is running.

**Communication Between Subsystems**

The primary flow of communications between differing subsystems, via IOMan, should happen in a high-to-low ring level permissions curve. In this context, it means that requests and communications should primarily be initiated in an environment with limited privileges (such as Ring 3), and transition or propagate to an environment where the privileges afforded are enough to complete the request and nothing more (Rings 2, 1, or 0). Then, the flow of execution must return to the entity that requested the access. Elevation should be short and purposeful, and requestors should not overstay their welcome in the lower ring levels. The only exception to this rule is for supervisor purposes, if Super ever has to resurrect a fallen Subsystem.
A System Architecture, for an Insignificant OS

More on OS Spaces/Modes

As mentioned above, the OS has been divided into three unique spaces, in opposed to two (unlike most modern OS’s). It is not possible to run this OS without SystemSpace. It is this way for security reasons. Here are some of the intended characteristics for the Spaces mentioned:

**KernelSpace**
- Ring(s): 0
- Mode(s): real mode
- Memory/Address Space: physical, shared
- Managing Entities: Super; FSMan (reduced functionality for loading modules*)

**SystemSpace**
- Ring(s): 1 and 2 (depending on the operation being performed)
- Mode(s): protected mode; 'huge real' mode (rare)
- Memory/Address Space: virtual, shared; with basic protections
- Managing Entities: IOMan; ResMan; MemMan; DevMan

**UserSpace**
- Ring(s): 3
- Mode(s): protected mode (32-bit); long mode + compatibility mode (64-bit OS)
- Memory/Address Space: virtual, isolated
- Managing Entities: RTMan; FSMan; UXMan

* Includes Subsystems, Drivers, and other system modules that rely on a reduced filesystem

As mentioned above, system modules may rely on a reduced filesystem of sorts. For information on why this is necessary please see the following:
- OS Dev.org: [Modular Kernel (Kernel Designs)](https://osdev.org/)
- OS Dev.org: [Hybrid Kernel (Kernel Designs)](https://osdev.org/)

More changes are possibly on the way, but these are the ones that I've decided to record for now.