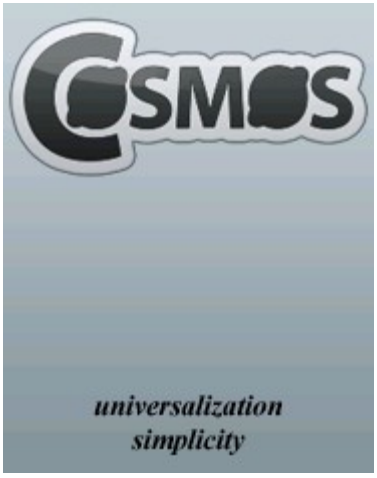
Cosmos Design Version .01(Draft)



Writtem by **Anarky** anarky444 (At) gmail.com

Lets Start with Something.

Basic Design Requirements

We will like to design Cosmos as a

- as a reliable OS which never hangs(of course until hardware fails).whichever program crashes, the OS should never hang or go unresponsive.
- is very safe(Safe in the sense which is free from buffer overflows,heap overflows,exploits etc.)
- Drivers/Programs should be verifiable. Even the kernel should be verifiable at a later stage(maybe using TALx86)
- as the LINUX of Tomorrow.

Initial Kernel Design Draft (based on what has already been done)

Kernel will be designed in 5 different parts (as opposed to only 2 in current generation kernel and drivers)

1. **Kernel Core**
    1) Interrupt Dispatcher
    2) Physical Memory Manager
2. **Kernel Runtime**
    1) Paging (it would also include a Per node SLUB allocator). Every thing below this stage should use this Memory Manager
    2) Process/Thread managing code
    3) File systems Read only driver(FAT, EXT whatever else is required)
    4) Messaging System.
3. **Kernel Bridge (ABI) :-** ABI Interface accessible to the mono runtime. This would only use Kernel runtime and thus cosmos can run on different architectures once Kernel Runtime has been ported to it.
4. **Mono Runtime:-** Mono Runtime which only uses the Kernel Bridge ABI to run. This is the last where IL2CPU will be used. Everything else including Drivers would be written in safe MSIL and will be interpret by the runtime.
5. **Drivers** (written in MSIL, would be run by Mono runtime/ together with features such as DMA access, Port access. Drivers will be nothing more than a program owning some special resources such DMA, I/O Ports etc

NOTE:- the Drivers part will be interpreted by the mono runtime and not by the IL2CPU

TODO!!! Where to put Virtual Memory, in the runtime or implemented as a driver ????

Other Note Worthy ideas

The OS should also consider the following stuff worthy in design

Cosmos should be a Single Address Space(SAS) OS. Since all the programs which we run will be verifiable we don't memory protection at all.(we won't allow pointer based C# programs)

This has numerous benefits such as

- Easier memory management
- faster Process switching(currently it takes 1000's of cycles while this will let us do it in 100's)
- Faster process creation

however, it has downfalls also, (but still better than today's generation)

No process can be larger than 3.5 (approx) GB of memory, Current generation OS (linux/ Windows) cannot run programs larger than 2 GB (under certain condition(such as PAE) it can be 3 GB)

Considering Drivers as Program.

Current generation driver model doesn't expect drivers to stop abruptly, however we will consider them as programs, so even a driver fails(in cosmos , it can never BSOD), we can start it again,(but some information might be lost in such case).and thus the system can continue without requiring a reboot

Memory Manager.

We expect to use a 3-tier architecture for memory allocator/deallocator , . This is because we are using MSIL interpreted/compiled code which is using GC. So we really expect a large no. of memory requests from the runtime. will implementing one more SLUB allocator level in the Mono runtime, cause any benefits, lets keep this question in place, and reach this stage, then we can have a lengthy discussion about this.

Current design has no place for Virtual Memory. It only considers Physical Memory

The structure we will use is

SLUB based per node allocator

Paging ,

Physical memory allocator