TBOS v1.0: Foundations

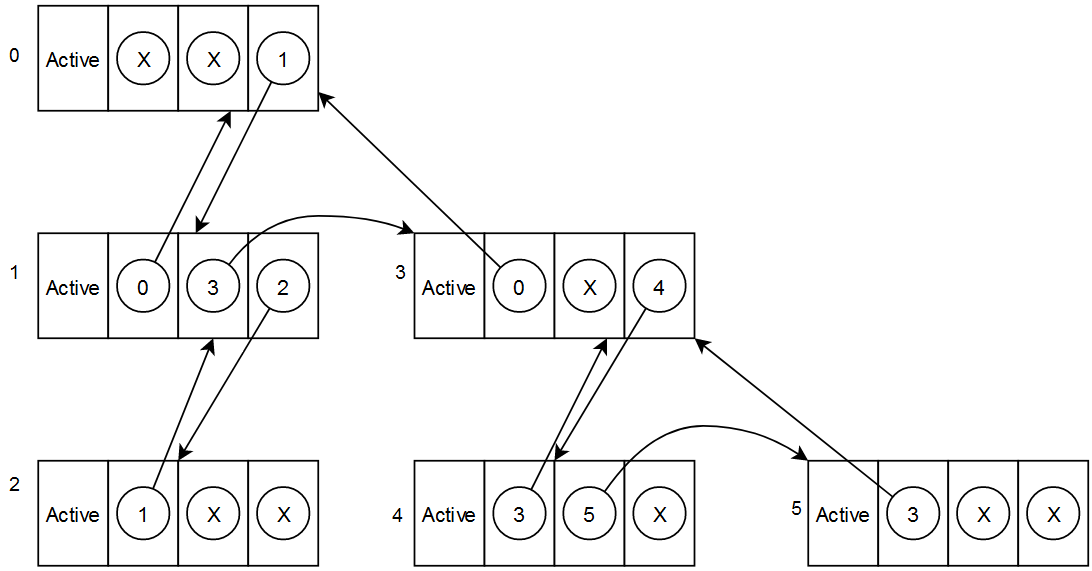
Telescope Basic Operating System v1.0

TBOS v1.0 is the “Foundations” version because it provides the foundations that future versions will use to achieve the following goals, as outlined in the Need-To-Know document:

- Multiple supported syscalls
- Loading programs from anywhere in a given drive.
- Spawning child threads with given properties
- Support for both custom semaphores and I/O semaphores
- Simulated I/O device interrupts & wait for I/O
- Support for all 4 CPU interrupts
- Support structures for processes

Until then, this is what TBOS does:

- Support for 16 maximum processes
    - Process metadata is split into a StateList and a ProcList
        - ProcList contains the PCB’s from PandOS, but each one starts with a “main word” which contains all the immediately important information about the process, like its children, parent, and sibling, as well as whether it’s active or not.
        - StateList contains 12 registers per state, excluding IO, PC, EXA, and CTD.
    - Each process’ ID is its index in these two lists.
- Process relations are maintained via a parent, child, and next sibling pointer.
    - Proc 0 is used as a null pointer, given that it is the only process which has no parent and no siblings, and it cannot be a child of anything.
    - Here’s an example graph of how these pointers work:



- This is how it is understood by the OS, but these actually exist in one continuous array in memory along with all of the inactive processes.

- Allocates memory using a 4-word bitmap, where each bit represents a 1KW “chunk” of addresses.
  - Memory is allocated in the next open space of a given size.
- The OS has 3 main functions:
  - All syscalls are treated as a request to spawn a process from a program in the drive pointed to by FLG. This program is assumed to be at address 0 in the drive. This program must also be in .tload format. If there are any issues, an error code in the form of 0xF00X will be placed in the OUT register.
    - For allocation issues, X=0
    - If the OS has reached its process limit, X=1
    - If it’s the kernel-only drive 0, X=2
    - If the FLG is not set to a drive, X=3

    The thread for this program is created as a child of the current thread. Furthermore, control is given back to the current process after this and is not charged to its CTD.
  - When CTDZ occurs, the next process is run by the dispatcher.

        - The next process is the next active process up from CurrentProcID in the ProcList.
    - When OOB or PRIV occurs, the current process is terminated.
        - This also terminates all of its children processes.
    - The OS does not support any of the 4 interrupts currently (though it technically supports STOP). If an interrupt occurs, the OS errors and halts.
- Additionally, the OS has some helpful interactions with the Telescope Computer
    - It sets the computer status code depending on what is happening
        - 1 = The OS has started. This also causes a breakpoint in the simulation
        - 2 = Root has been started.
        - 600D = All processes have terminated successfully.
        - FF00 = Bad startup error
        - FF01 = Unsupported exception type error.
    - The OS starts the system wide cycle counter right after status code 1
        - It stops the cycle counter upon halting.

Here are some values in the TBOS.tasl source code that can be changed to have different effects:

> @CTD_VAL is the value that CTD is set to every time. As of writing, this is very small for testing, but also because processes currently do not have access to some kind of "wait" syscall. This lets processes enter into an infinite waiting loop, while also letting other processes execute their tasks in a reasonable amount of time.
>
> @CHUNK_SIZE and the three lines below it can be used to configure memory allocation. Just make sure to follow the rules for setting these.

Important: @MAX_PROC_COUNT cannot be increased without setting the (many) masks and shifts below it to something which support a larger number of processes. The maximum these could support is 32.