Joshua Bennett
2024-05-03

webOS Technical Report

webOS is a Linux-based operating system developed by Palm and first released in January 2009 as part of the Palm Pre. Palm was then acquired by HP in April 2010, after which, in February 2011, it announced the intention to use webOS for all of its devices in the future. This included a few smartphones and the HP TouchPad. However, by December 2011 the TouchPad was abandoned and in August 2012 the source code for webOS was released. Following this, on February 25, 2013, LG purchased webOS for use in smart TVs, later expanded to a lot of other devices. This includes smart watches, fridges, and display panels. Later, an open source version of webOS was released on March 19, 2018. Since it is open source, this version is easy to analyze, so it is what this paper will be based on.

webOS has two types of apps: web app and native app. Native apps are handled by the linux kernel, beyond being launched and ended by the System and Application Manager. Web apps are managed by the Web Application Manager, and therefore have a lifecycle used by it. They have three states: Launched, Not Launched, and Suspended. This is very different from what we learned about in class, probably because this is a mobile OS and not a computer OS. See Figure 1 for an overview of the events and states of webapps.

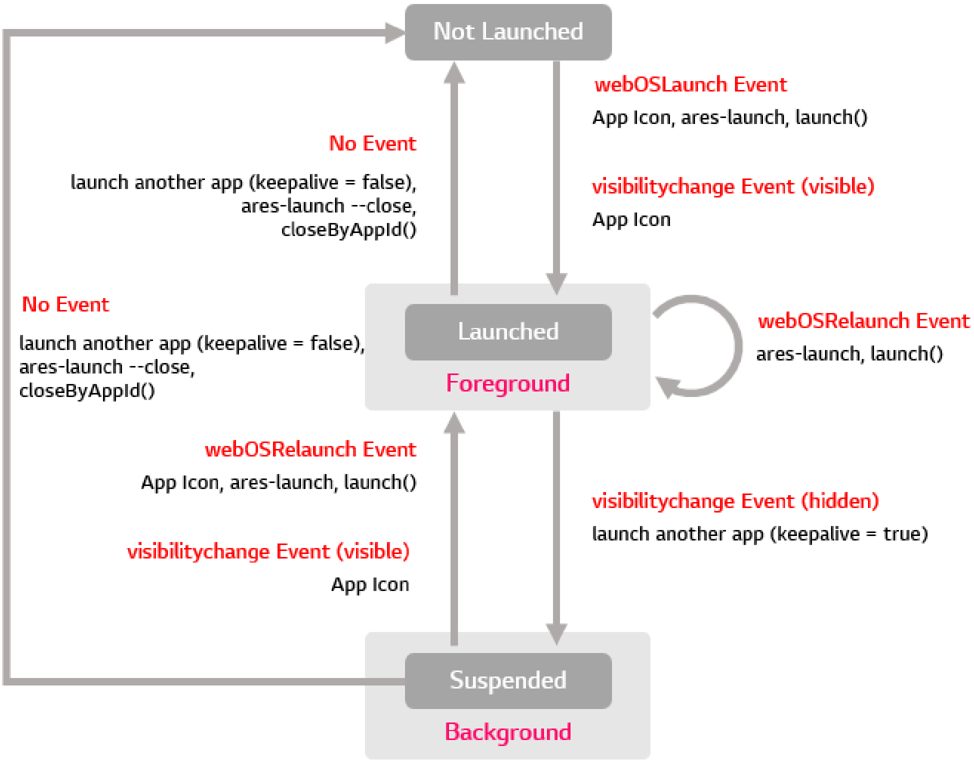
Not Launched
webOSLaunch Event
App Icon, ares-launch, launch()
visibilitychange Event (visible)
App Icon
No Event
launch another app (keepalive = false),
ares-launch --close,
closeByAppId()
No Event
launch another app (keepalive = false),
ares-launch --close,
closeByAppId()
Launched
Foreground
webOSRelaunch Event
ares-launch, launch()
webOSRelaunch Event
App Icon, ares-launch, launch()
visibilitychange Event (visible)
App Icon
visibilitychange Event (hidden)
launch another app (keepalive = true)
Suspended
Background

Figure 1: Web App Lifecycle (from https://www.webosose.org/docs/guides/core-topics/application-management/web-app-lifecycle/)

The Linux kernel has its own process management with states for waiting, ready, and running. This is very similar to what we learned in class, which makes sense since it is a very common OS and we even discussed it specifically a few times. See Figure 2 for an overview of the process (task) states in Linux.

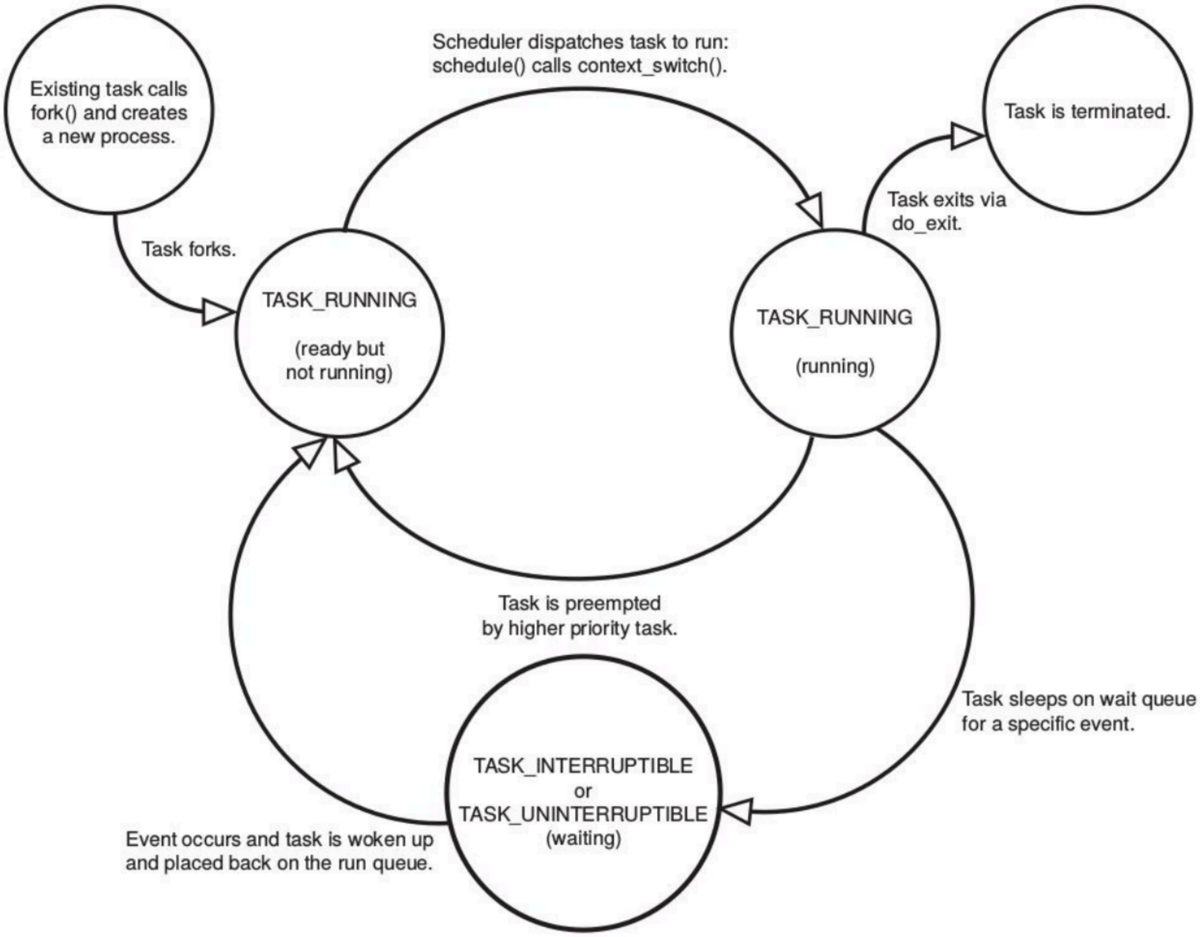
Existing task calls fork() and creates a new process.
Scheduler dispatches task to run: schedule() calls context_switch().
Task is terminated.
Task exits via do_exit.
Task forks.
TASK_RUNNING (ready but not running)
TASK_RUNNING (running)
Task is preempted by higher priority task.
Task sleeps on wait queue for a specific event.
TASK_INTERRUPTIBLE or TASK_UNINTERRUPTIBLE (waiting)
Event occurs and task is woken up and placed back on the run queue.

Figure 2: Linux task lifecycle. (from https://lass.cs.umass.edu/~shenoy/courses/spring20/lectures/Lec09.pdf*, page 29)*

As webOS is built from the Linux kernel, it uses the Linux scheduler. As discussed in class, Linux used to use the O(1) scheduler, but it was replaced with several options. The new default, the Completely Fair Scheduler (CFS) was introduced in 2007. While I don't know for sure, I'm going to assume that webOS uses the default, CFS. The CFS tries to emulate an ideal multiprogramming CPU, which gives every process an equal share of CPU power (see Figure 3).

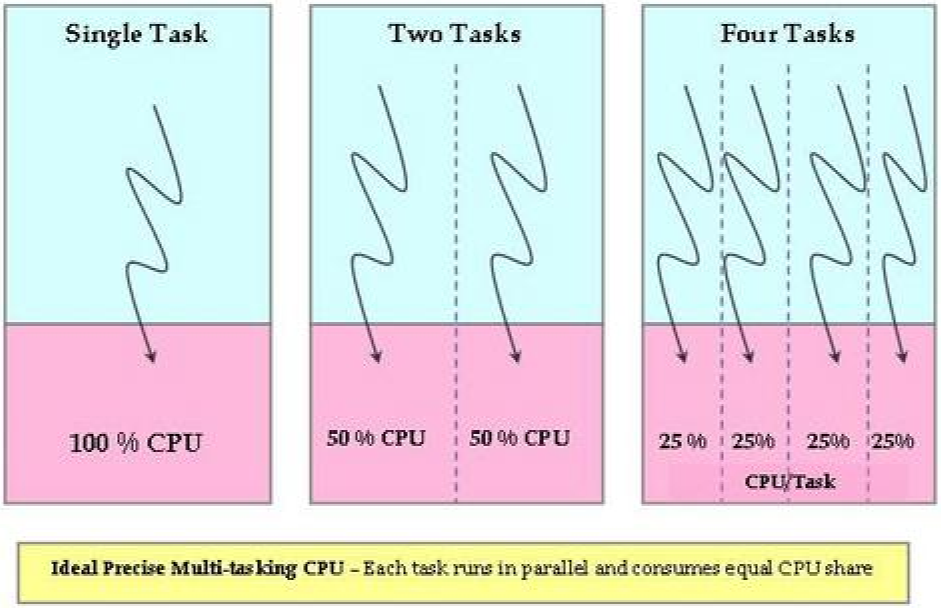
Single Task
Two Tasks
Four Tasks
100 % CPU
50 % CPU
50 % CPU
25 %
25%
25%
25%
CPU/Task
Ideal Precise Multi-tasking CPU – Each task runs in parallel and consumes equal CPU share

Figure 3: An ideal multitasking CPU (from https://www.linuxjournal.com/node/10267*)*

This is done by, for every waiting process, tracking the amount of time it would've had on the CPU, then always attempting to run the process that has missed out on the largest amount of CPU time. The processes are stored in a red-black tree which is keyed by this tracker. As the name implies, this policy values fairness above all else. Therefore, it does not discriminate on CPU vs. IO-bound processes. It is worth noting that Linux uses preemptive scheduling, so the currently running process does not need to exit or do IO to be replaced. Since the processes are prioritized based on wait time, starvation is impossible. However, it means that an IO heavy process may need to wait a bit after completing its IO to be allowed back on the CPU.

For memory management, the OS layer is handled entirely by the Linux kernel. The Linux kernel uses paged memory because it makes administration of memory easier. It is basically exactly like we learned in class, but it has three levels of page tables. Here, the first two page tables lead not to the page where the data is but the page where the next page table is. See Figure 4 for how this looks in practice.

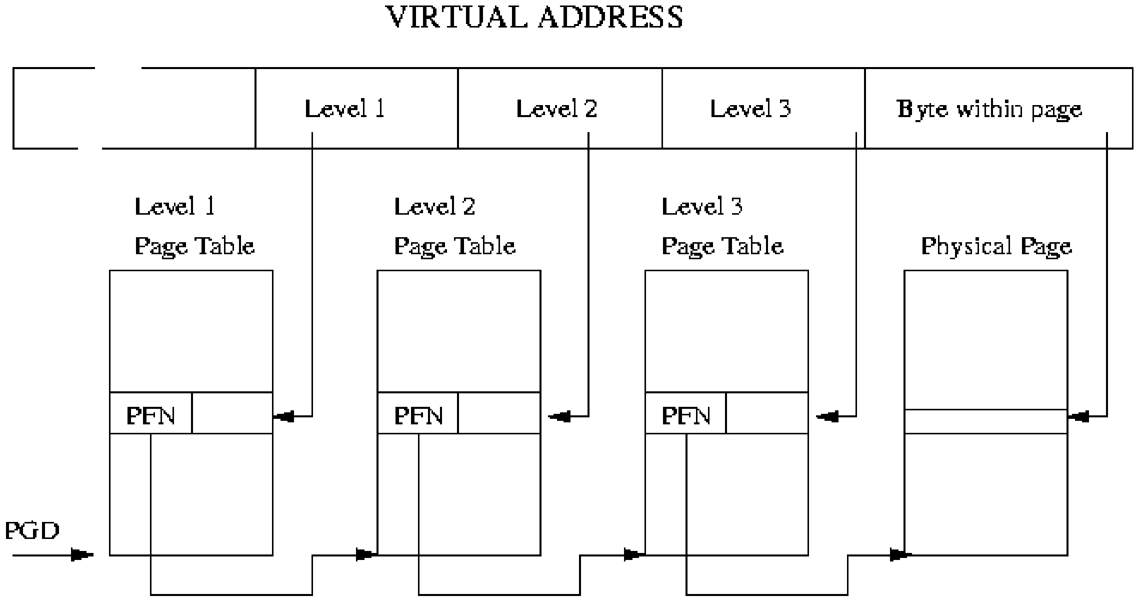

Figure 4: Three Level Page Tables (from https://tldp.org/LDP/tlk/mm/memory.html*)*

On the process side, code will call malloc and related calls to allocate memory, which is provided by a version of jemalloc, which is the allocator used in FreeBSD.

One interesting feature of webOS is the interface. It uses a card-based multitasking system with gesture navigation, which will be familiar to anyone who uses an iPhone or android device. Interestingly, many of the features of the interface were in webOS first, before being added to other mobile OSs. For example, swiping up from the bottom to open first a quick bar then the app cards then the home screen like recent iPads work is very similar to how webOS worked on TouchPads, despite the time difference.

Works Cited

"Architecture Overview." *webOS Open Source Edition*, https://www.webosose.org/docs/guides/core-topics/architecture/architecture-overview/#system-and-application-manager. Accessed 3 May 2024.

Bohn, Dieter. "What the iPhone X Borrowed from the Palm Pre." *The Verge*, 15 Sept. 2017, https://www.theverge.com/2017/9/15/16300402/iphone-x-webos-palm-pre-cardsgestures-nostalgia. Accessed 3 May 2024.

Contributors to Wikimedia projects. “WebOS.” *Wikipedia*, 26 Apr. 2024, https://en.wikipedia.org/wiki/WebOS. Accessed 3 May 2024.

“Manual Page.” *Jemalloc*, https://jemalloc.net/jemalloc.3.html. Accessed 3 May 2024.

Rusling, David A. “Chapter 3: Memory Management.” *The Linux Kernel*, https://tldp.org/LDP/tlk/mm/memory.html. Accessed 3 May 2024.

Shenoy, Prashant. “A Deep Dive into the Linux Kernel Processes and Syscall.” *Laboratory for Advanced System Software*, University of Massachusetts, https://lass.cs.umass.edu/~shenoy/courses/spring20/lectures/Lec09.pdf. Accessed 3 May 2024.

Singh Pabla, Chandandeep. “Completely Fair Scheduler.” *Linux Journal*, 1 Aug. 2009, https://www.linuxjournal.com/node/10267. Accessed 3 May 2024.

“Web App Lifecycle.” *webOS Open Source Edition*, https://www.webosose.org/docs/guides/core-topics/application-management/web-app-lifecycle/. Accessed 3 May 2024.

“webOS Developer.” *LG Developer*, LG, https://webos.developer.lge.com/. Accessed 3 May 2024.

webosose. “GitHub - Webosose/Jemalloc: Jemalloc Used in webOS.” *GitHub*, https://github.com/webosose/jemalloc. Accessed 3 May 2024.