A new boot process for Plan 9

Iruatã Souza

4th IWP9
October 22, 2009

`http://iru.oitobits.net`
`http://src.oitobits.net/9null`

What we will see

Motivation

Plan 9 on PC
- BIOS and MBR
- Primary Boot Sector (PBS)
- *9load(8)*
- Stock kernels
- *boot(8)*

(Re)writing the boot
- pbs32.s
- 9pcload
- new *boot(8)*

Motivation

To avoid unneeded maintenance efforts

Simplification

Generalization

Plan 9 on PC – BIOS and MBR

BIOS
- Power On Self Test
- Read disk sector 0 to 0x7C00 (physical)
- Jump there

Typical MBR (disk sector 0)
- Relocates itself
- Parse master partition table
- Find active (bootable) partition
- Read partition sector 0 to 0x7C00 (physical)
- Jump there

Plan 9 on PC – Partition Boot Sector (PBS)

- Resides on each partition’s sector 0

- Read Plan 9 partition table

- Search 9fat for a file called ’9LOAD   ’ (8.3 format)

- Use interrupt 0x13 to read 9LOAD to 0x1000 (physical)
  – 16bit segmented: limited to reading ˜1MB of data

Plan 9 on PC – *9load(8)*

- Setup APM

- Setup VGA

- Enable 32bit protected mode

- Load boot configuration

- Load a kernel

Plan 9 on PC – *9load(8)*

Loading boot configuration

```
search plan9.ini or plan9/plan9.ini
if found
    read at most 100 key=value lines
if not
    ask for a kernel to load
store configuration at CONFADDR (0x1200)
```

no way to set configuration at runtime

Plan 9 on PC – *9load(8)* (cont’d)

Finding and loading a kernel

```
  use boot media routines to find a FAT partition
  find the $bootfile kernel in the given partition
  load it to 0x1000 (physical)
  jump to 0x1000
```

kernel **must** reside on FAT

kernel can be gzip compressed

Plan 9 on PC – *9load(8)* **(cont'd)**

Separate source code tree

Existing Plan 9 features (filesystem support, device drivers, &c) must be ported to *9load(8)* in order to be used to boot a kernel

PXE support included

Plan 9 on PC – Stock kernels

Expect to be loaded by *9load(8)*
  - to name one, *sd(3)* expects partitioning information
    to be stored in a CONFADDR line

Can live reboot into other kernels using *reboot(8)*

Plan 9 on PC – *boot(8)*

First user program to run

> `connect to file server (specified by` *plan9.ini(8)*`)`
> `mount file server as the namespace root`
> `run` *init(8)*

It does so by execing user programs (*factotum(4)*, *fossil(4)*, &c)

Written in C

(Re)writing the boot

Russ Cox did solve part of the problem with his load program.
It still left us:

- the need for *plan9.ini(8)*
- the need for kernel and *plan9.ini(8)* to be on FAT
- the need for local root to be *kfs(4)* or *fossil(4)*

9null is the effort under which a new PBS, new *boot(8)*,
and kernel configuration are being written

(Re)writing the boot – pbs32.s

- enable 32bit protected mode

- read disk sectors in sequence until an *a.out(6)* header is found

- read the *a.out(6)* file to 0x00100000

- jump to 0x00100020

(Re)writing the boot – pbs32.s (cont'd)

- uses ATA commands to read sectors
  – only tested with hard disk drives

- file must be on contiguous blocks
  – as in 9fat

- do not handle configuration

(Re)writing the boot − New kernel configuration

9pcload

9pcf plus usual shell tools (*rc(1)*, *awk(1)*, *sed(1)*, *&c*)

Tells *boot(8)* if it is the kernel being loaded

(Re)writing the boot – New *boot(8)*

```
if loaded by 9pcload
  ask for kernel to load (method!fspath!kernel)
if not
  ask for root file server (method!path)

if answer is ’!rc’
  run rc(1)

if the prompt timedout
  read plan9.ini(8) to memory

reboot $bootfile
```

(Re)writing the boot – New *boot(8)* (cont'd)

- both interactive and batch (with equal syntax)

- allows for experimentation with unusual boot scenarios

- mostly written in *rc(1)*

(Re)writing the boot – Execution paths

9load(8)

`BIOS -> mbr -> pbs(lba).s ->` *9load(8)* `-> final kernel -> boot(8) -> init(8)`

9null

```
BIOS -> mbr -> pbs32.s -> 9pcload -> boot(8) ->
final kernel -> boot(8) -> init(8)
```

Conclusion

To avoid unneeded maintenance efforts

- no need to port from kernel/user to *9load(8)*
- user may experiment with boot configurations without the need for a file

Simplification

- *rc(1)* seems a more natural fit for coordinating programs
- standard tools can be used in the boot process normally

Generalization

- access to the full range of Plan 9 services while booting
- *boot(8)* is closer to the other system programs

Future Work

- Testing in (un)usual situations

- PXE