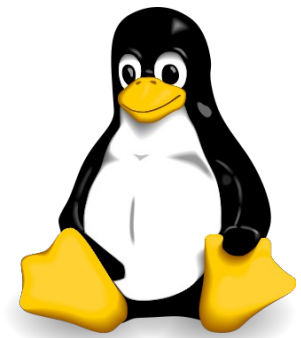


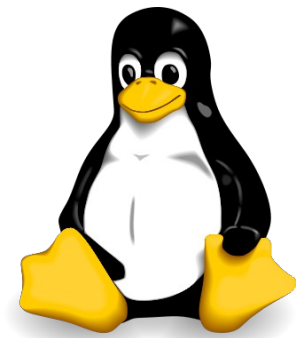
---

# More Kernel Bits



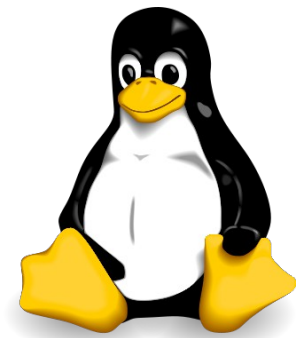
# Rebooting

- Why would one ever need to reboot on Linux?
  - Installing new driver?... (No)
  - Installing new applications?... (No)
  - Installing new.... Anything?... (No)
  - Reconfiguring.... (No)
  - Updating.... (No)



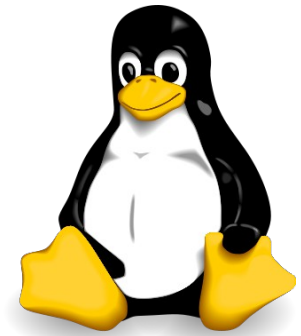
# Rebooting

- The only legitimate reason to reboot a kernel is for new kernel features (or bug fixes).
  - Modules take care of the rest
- But... What if we could reload a kernel without rebooting?
- Would you ever need to reboot?



# Reloading your Kernel

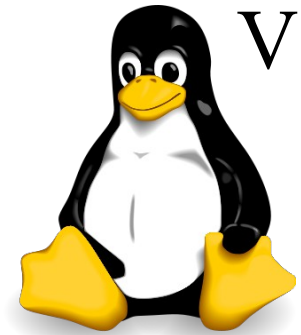
- Step 1:
  - Load a new kernel
  - Kexec does this part
  - Kexec, a user space utility
    - Sadly not easy to do right now, but possible and may become easier in the future.



# Reloading your Kernel Step 2

- Step 2:
  - Swap Kernels
  - We can do this with a (very) recent kernel feature called “relocatable kernel” allows us to Kexec to a non standard location.
- The idea: Fire up new kernel in location X, done with kexec and relocatable, then “switch over”

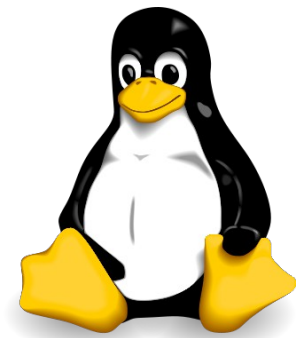
Voila, no reboot!



# Bootup

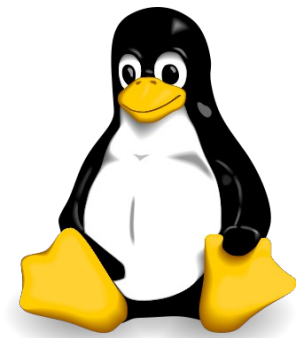
---

From GRUB to Gourmet Meal



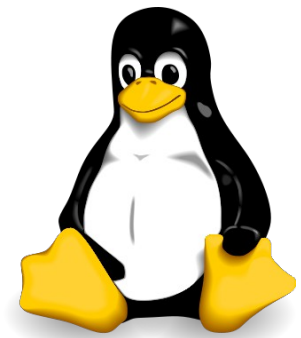
# Step 1 : Power button

- Power button initiates a ton of hardware processes.
- All of them begin with 'BIOS' (**B**asic **I**nput **O**utput **S**ystem)
- BIOS does a POST (**P**ower **O**n **S**elf **T**est)



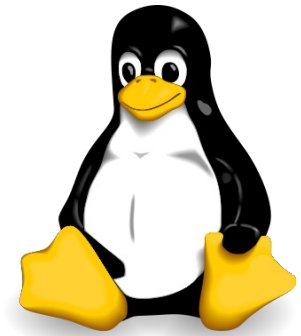
# B.I.O.S.

- Motherboard loads the BIOS from the BIOS chip and into the lowest 1MB of memory.
- BIOS is the lowest level of code on the system that controls the system as a whole .
  - a.k.a. Firmware is lower but it doesn't count.
- BIOS is also configurable – has its own menu system.



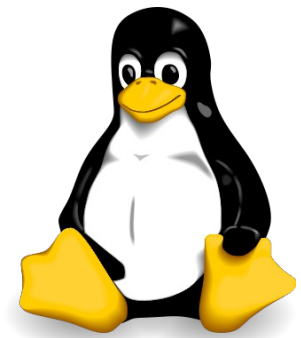
# POST

- A series of operations run by the BIOS.
- Tests that the necessary hardware is available. If something is obviously broken it attempts to report the issue (often beep codes).
- Does an integrity check of the BIOS itself!



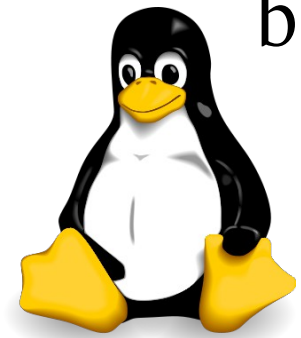
# Boot

- BIOS initializes all hardware, and then looks for a 'boot device'.
- Boot is short for bootstrap, the computer pulls itself up by its bootstraps.



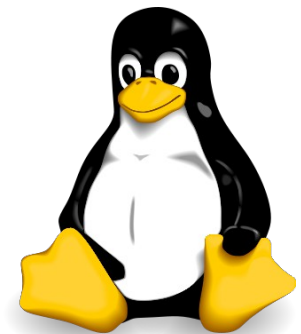
# Selecting a boot device

- BIOS has a list of devices to try and boot off of (CD-ROM, hard disks etc.) It tries them sequentially until one is found.
- On a hard disk, the first couple of sectors contain an MBR (**M**aster **B**oot **R**ecord) with information for boot.
- On Windows, this points straight to the Windows boot loader.



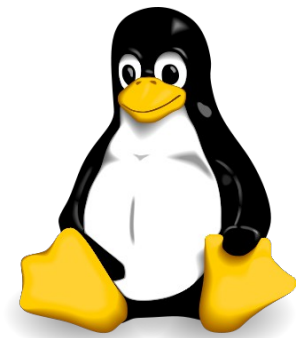
# GRUB : The Linux Boot Loader

- GRUB is a program which loads at boot and allows you to select a 'boot configuration' to run.
- You configure GRUB with the location of a kernel, and other boot options and grub will allow you to run these at system boot.
- GRUB can also load other bootloaders (Windows, OS X etc.)



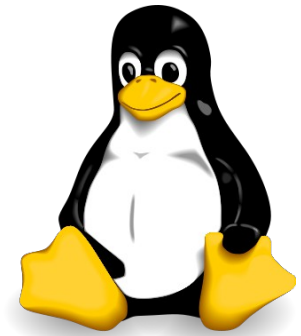
# A pre-OS OS

- GRUB provides an actual shell you can use before even booting an OS!
- This means GRUB must perform many of the functions of an OS (memory management etc.) and it does!
- Hence, Grub is awesome.



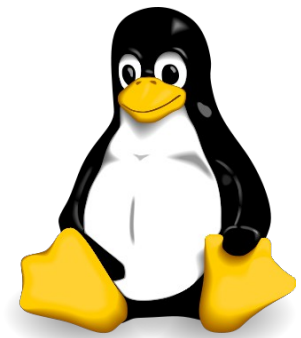
# Starting the Kernel

- GRUB then loads the kernel into memory and executes it.
- The kernel boot process could have a whole course of its own as it does a lot of things. On a simple level, it initializes all of the hardware drivers and sets up userspace.



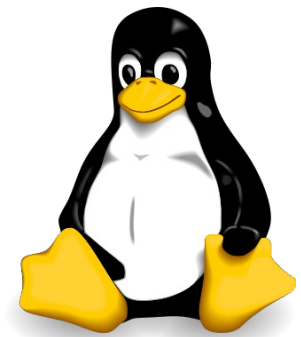
# Init [ OK ]

- The kernel will then run an 'init' system. The most common one, sysvinit you see on almost every linux computer.
- Sysvinit is the program which prints  
Starting also sound .... [OK]



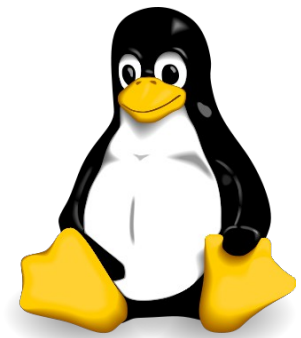
# Init lasts forever

- The init system sits in the background and manages running services in userspace.
- Responsible for shutting down these services on reboot / shutdown.
- Controls the system “runlevel”.
  - Runlevel 0 = shut down
  - Runlevel 1 = single user
  - Runlevels 2-5 = running (different sets of services)
  - Runlevel 6 = reboot



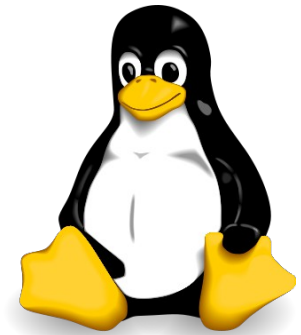
# Init Runlevels

- Each runlevel has an associated list of init scripts which should be started and additional actions.
- Examples (approx.)
  - Runlevel 3 – full system without `/etc/init.d/networking`
  - Runlevel 5 – all init scripts
  - Runlevel 0 - no init scripts + shutdown command



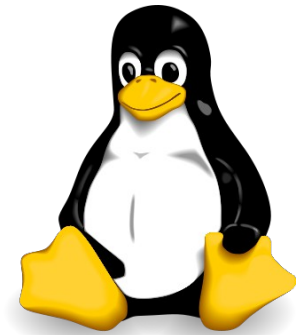
# XDM / GDM

- One of the last programs init starts up is called XDM (more commonly, GDM).
- XDM starts an X server and then launches a login manager like GDM.
- GDM is a login manager (others include XDM, KDM, Entrance etc.)
  - GDM is the place where you type in your username and password.



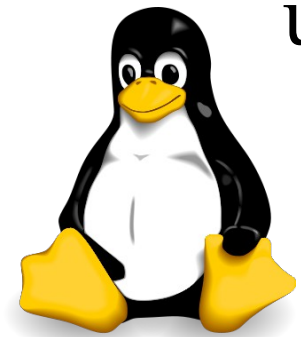
# To the window manager

- Once you login successfully GDM begins a 'session'.
- A session consists of a user logging in and running a session script.
- Session scripts most commonly execute some applications and a window manager (e.g. Gnome and Metacity).
  - “Startup applications” can be run in these scripts (~/.xinitrc, ~/.xsession etc.)



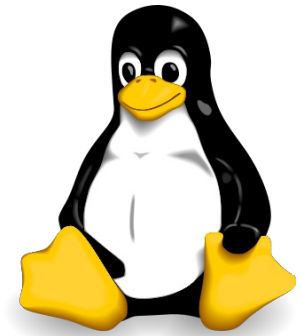
# Virtual Terminals

- The last very process that init starts is some form of 'GETTY' (commonly *agetty*)
- The gettys manage the systems virtual terminals.
- A system has 12 virtual terminals, 1->12.
- They are accessed via the key sequence ctrl+alt+F#
- Getty starts a terminal login on 1 through 6. X11 usually starts on 7.

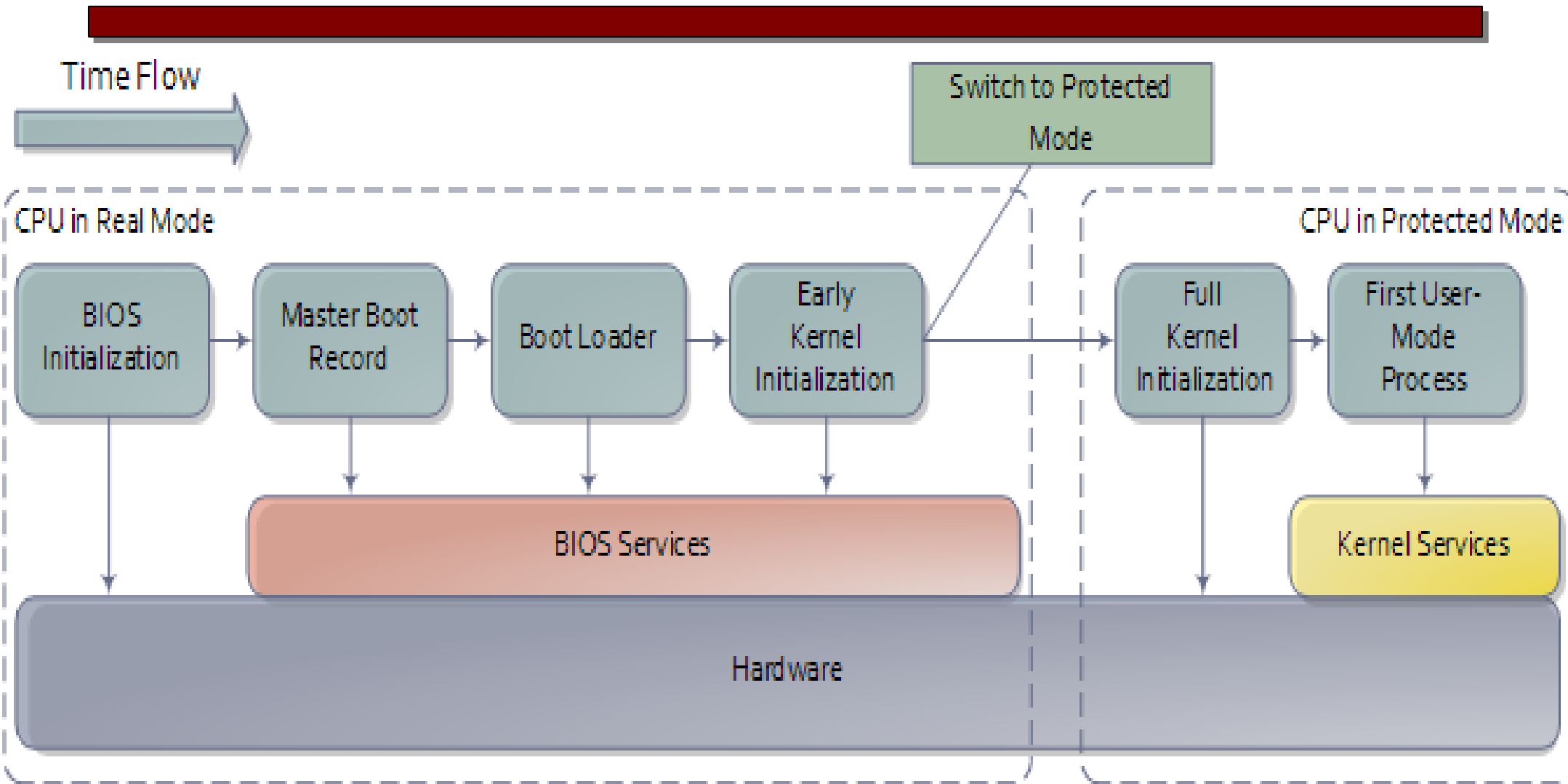


# Booted System

- You have several virtual terminals open, and an x session running on one of them.
- Play with your window manager indefinitely: that's a computer.



# Boot process



# Extra Interesting

- UPnP / DLNA
  - MediaTomb
- Kernel is more than 30% drivers!

