Understanding User and Kernel Mode

- Site Reliability Engineer HandBook
- Introduction
- Programming Language
    - Python
        - Time Format
        - Subprocess
        - Multiprocess
        - Rename
        - SMTP
        - Single instance of program
        - Argparse
        - Requests
        - Pyinstaller
        - Readlines
        - Raw Input
        - With Open
        - Configparser
        - Gzip
        - Listdir
        - Basename
        - Dirname
        - Traversing a Directory Tree
        - Startswith
        - Endswith
        - Virtualenv
        - Regular Expressions
        - Supervisor
        - Socket
        - Exception Errors
        - Raw_input
        - Threading
        - Unittest
        - Why is it better to use “#!\/usr\/bin\/env NAME” instead of “#!\/path\/to\/NAME” as my shebang?
        - OS
        - Decorator
        - String Formatting

        - SimplePrograms
        - 'all', 'any' are Python built-ins
        - TemporaryFile
        - How to capture stdout in real-time with Python
        - Python simple techniques and common reference
        - python reference fragments
        - getpass
        - Method overriding in Python
        - Multiple levels of 'collection.defaultdict' in Python
        - String Format
        - Logging
        - Convert Unicode Object to Python Dict
        - The dir( ) Function
        - Python dictionary has_key() Method
        - glob – Filename pattern matching
        - Lambda, filter, reduce and map
        - doctest – Testing through documentation
        - Load Python code dynamically
        - Map, Reduce, Zip, Filter
        - DICTIONARY COMPREHENSION
- Linux Command Line Tool
    - Basic
    - DIFF
    - AC
    - AWK
    - CHMOD
    - NMAP
    - NETSTAT
    - Flock
    - Traceroute
    - FIND
    - GREP
    - Crontab
    - Kill
    - SED
    - CUT
    - CURL
    - IFCONFIG
    - TCPDUMP
    - TAR
    - LSOF

- SORT
- Xargs
- Iptables
- xargs vs. exec {}
- Hdparm
- UNIQ
- STAT
- Execute Commands in the Background
- TAIL
- WGET
- Date
- FDISK
- Mount
- Make SWAP File
- Create a New User
- Create a New Group
- Setup SSH Passwordless Login in OpenSSH
- Parted
- RSYNC
- YUM
- RPM
- APT
- Install from Source
- Log Rotate
- FREE
- DF
- DU
- Sysctl
- NICE
- Renice
- PS
- DD
- BC
- LDD
- getcap, setcap and file capabilities
- Linux_Basename
- PMAP
- Alternative
- Readlink
- logrotate
- PIDOF

    - Dmidecode
    - lshw
    - printenv
    - SS
    - w
    - Strace
    - pstree
    - USERMOD
    - ltrace
    - ethtool
    - IP
    - Sar
    - nethogs
    - zip
    - FPM
    - getent
    - ipmitool
    - Building RPMs
    - Megacli
        - Megacli package version
    - RKhunter
    - fping
    - blkid
    - FSCK
    - Package Manager
    - mktemp
    - ls
    - Comm
    - taskset
    - fio
    - tree
    - ARP
    - lsblk
- How-To
    - CentOS: nf_conntrack: table full, dropping packet
    - How To Fix “Error: database disk image is malformed” On CentOS \/ Fedora
    - Finding the PID of the process using a specific port?
    - How-To create hashed SSH password
    - How to display and kill zombie processes
    - Shell command to bulk change file extensions in a directory (Linux)

- 8 Powerful Awk Built-in Variables – FS, OFS, RS, ORS, NR, NF, FILENAME, FNR
- Changing the Time Zone
- HOW DO I DISABLE SSH LOGIN FOR THE ROOT USER?
- How-To rename the extension for a batch of files?
- How-To disable IPv6 on RHEL6 \/ CentOS 6 \/ etc
- How to clear the ARP cache on Linux?
- How-To crontab running as a specific user
- Ansible – exclude host from playbook execution
- HOWTO: Use Wireshark over SSH
- How-To Change Network Interface Name
- How-To Creating a Partition Size Larger Than 2TB
- Hot-To Linux Hard Disk Format Command
- Hadoop Troubleshooting
- Hive Troubleshooting
- HowTo Set up hostbased authentication for passphraseless SSH communication.
- Difference between a cold and warm reboot
- ls -l explained
- df falsely showing 100 per cent disk usage
- FSCK explained
- Manually generate password for \/etc\/shadow
- How To Change Timezone on a CentOS 6 and 7
- Setting ssh private key forwarding
- Persist keys in ssh-agent on OS X
- SSH Essentials: Working with SSH Servers, Clients, and Keys
- How to Change JVM Heap Setting (-Xms -Xmx) of Tomcat – Configure setenv.sh file – Run catalina.sh
- SSH ProxyCommand example: Going through one host to reach another server
- How to get Linux's TCP state statistics
- Linux TCP retransmission rate calculation
- How to determine OOM
- How-to check Java process heapsize
- Troubleshooting network issues
- How to check what sudo acces a user has?
- How to copy your key to a remote server?
- Linux date and Unix timstamp conversion
- SSH client personalized configuration
- How to Error Detection and Correction
- How To Kerberos
- How to identify defective DIMM from EDAC error on Linux

    - Howto Install and Configure Cobbler on Centos 6
    - How To Use GPG to Encrypt and Sign Messages on an Ubuntu 12.04 VPS
    - HowTo: Debug Crashed Linux Application Core Files Like A Pro
    - Create init script in CentOS 6
    - Linux Change Disk Label Name on EXT2 \/ EXT3 \/ EXT4 File Systems
    - How to retrieve and change partition's UUID Universally Unique Identifier on linux
    - Using Text-Mode Serial Console Redirection
    - How to Write Linux Init Scripts Based on LSB Init Standard
    - How to create a Debian package
    - How to create a RPM Package
    - How to solve EDAC DIMM CE Error
    - How to solve fsck.ext4: Unable to resolve UUID\/LABEL
    - How to expand an existing LSI raid array using MegaCli
    - How to change user GID and UID in Ubuntu
    - How to read a segfault kernel log message
    - How to add cron job via command line
    - How to restrict process CPU usage using nice, cpulimit, and cgroups
- Data Structure
- Service
    - Cloud-Init
    - ETCD
    - RESTful API HTTP methods
    - Web cache
    - Mesos
    - ELK
    - Cassandra
    - Hive
        - Hive notes
    - Elasticsearch
    - Scylla
    - Zookeeper
- Automation Tool
    - Ansible
    - Salt
        - Salt use notes
- Networking Devices
    - Cisco
    - Juniper
- Version Control
    - SVN

- Editor
    - VIM
- Systems Performance
    - Common Performance Troubleshooting Tool

1. Kernel Mode

In Kernel mode, the executing code has complete and unrestricted access to the underlying hardware. It can execute any CPU instruction and reference any memory address. Kernel mode is generally reserved for the lowest-level, most trusted functions of the operating system. Crashes in kernel mode are catastrophic; they will halt the entire PC.

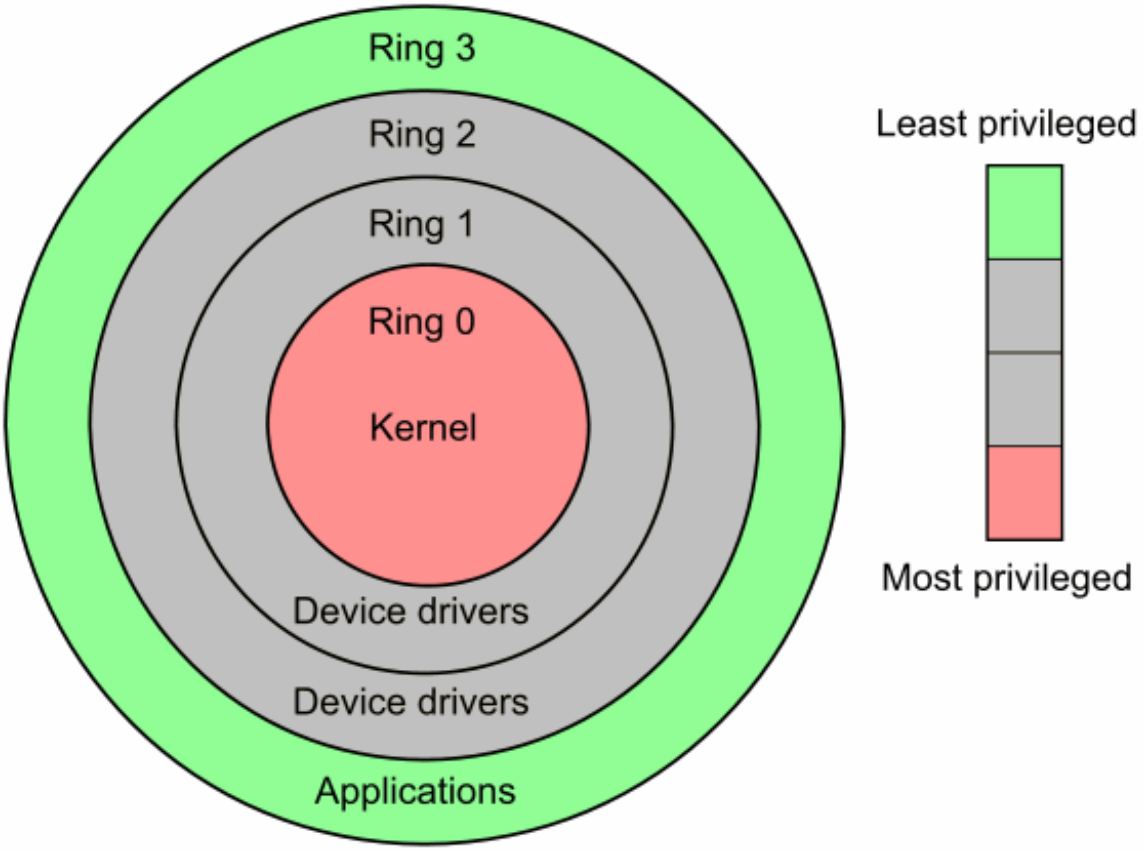
2. User Mode

In User mode, the executing code has no ability to directly access hardware or reference memory. Code running in user mode must delegate to system APIs to access hardware or memory. Due to the protection afforded by this sort of isolation, crashes in user mode are always recoverable. Most of the code running on your computer will execute in user mode.

It's possible to enable display of Kernel time in Task Manager, as I have in the above screenshot. The green line is total CPU time; the red line is Kernel time. The gap between the two is User time.

These two modes aren't mere labels; they're enforced by the CPU hardware. If code executing in User mode attempts to do something outside its purview-- like, say, accessing a privileged CPU instruction or modifying memory that it has no access to -- a trappable exception is thrown. Instead of your entire system crashing, only that particular application crashes. That's the value of User mode.

x86 CPU hardware actually provides four protection rings: 0, 1, 2, and 3. Only rings 0 (Kernel) and 3 (User) are typically used.



If we're only using two isolation rings, it's a bit unclear where device drivers should go-- the code that allows us to use our video cards, keyboards, mice, printers, and so forth. Do these drivers run in Kernel mode, for maximum performance, or do they run in User mode, for maximum stability? In Windows, at least, the answer is it depends. Device drivers can run in either user or kernel mode. Most drivers are shunted to the User side of the fence these days, with the notable exception of video card drivers, which need bare-knuckle Kernel mode performance. But even that is changing; in Windows Vista, video drivers are segmented into User and Kernel sections. Perhaps that's why gamers complain that Vista performs about 10 percent slower in games.

The exact border between these modes is still somewhat unclear. What code should run in User mode? What code should run in Kernel mode? Or maybe we'll just redefine the floor as the basement-- the rise of virtualization drove the creation of a new ring below all the others, Ring -1, which we now know as x86 hardware virtualization.

User mode is clearly a net public good, but it comes at a cost. Transitioning between User and Kernel mode is expensive. Really expensive. It's why software that throws exceptions is slow, for example. Exceptions imply kernel mode transitions. Granted, we have so much performance now that we rarely have to care about transition performance, but when you need ultimate performance, you definitely start caring about this stuff.

Probably the most public example of redrawing the user / kernel line is in webservers. Microsoft's IIS 6 moved a sizable chunk of its core functionality into Kernel mode, most

notably after a particular open-source webserver leveraged Kernel mode to create a huge industry benchmark victory. It was kind of a pointless war, if you ask me, since the kernel optimizations (in both camps) only apply to static HTML content. But such is the way of all wars, benchmark or otherwise.

The CPU's strict segregation of code between User and Kernel mode is completely transparent to most of us, but it is quite literally the difference between a computer that crashes all the time and a computer that crashes catastrophically all the time. This is what we extra-crashy-code-writing programmers like to call "progress". So on behalf of all programmers everywhere, I'd like to say thanks User mode. You rock!