PID Definition · Site Reliability Engineer HandBook

s905060.gitbooks.io/site-reliability-engineer-handbook/content/pid_definition.html

        - 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
- Storage
    - Object Storage
    - How an object store differs from file and block storage
- Monitoring
    - Nagios
    - Zabbix
    - Graphite
        - The architecture of clustering Graphite
- Database
- Algorithm
    - Insertion Sort
    - Hill Sort
    - Bubble Sort
    - Quick Sort
    - Directly Select Sort
    - Heap Sort
    - Merge Sort
    - Radix Sort
    - Cache algorithm definition
- Software Engineering
- 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
- Editor
  - VIM

Powered by **GitBook**

PID Definition

- PID: Process Id
- PPID: Parent Process Id (the one which launched this PID)
- TGID: Thread Group Id

A PID (i.e., process identification number) is an identification number that is automatically assigned to each process when it is created on a Unix-like operating system.

A process is an executing (i.e., running) instance of a program. Each process is guaranteed a unique PID, which is always a non-negative integer.

The process init is the only process that will always have the same PID on any session and on any system, and that PID is 1. This is because init is always the first process on the system and is the ancestor of all other processes.

A very large PID does not necessarily mean that there are anywhere near that many processes on a system. This is because such numbers are often a result of the fact that PIDs are not immediately reused, in order to prevent possible errors.

The default maximum value of PIDs is 32,767. This maximum is important because it is essentially the maximum number of processes that can exist simultaneously on a system.Although this will almost always be sufficient for a small system, large servers may require many more processes. The lower the maximum value, the sooner the values will wrap around, meaning that lower values do not necessarily indicate processes that started to run earlier.

In Unix-like operating systems, new processes are created by the fork() system call. The PID is returned to the parent enabling it to refer to the child in further function calls. The parent may, for example, wait for the child to terminate with the waitpid() function, or terminate the process with kill().

There are two tasks with specially distinguished process IDs: swapper or sched has process ID 0 and is responsible for paging, and is actually part of the kernel rather than a normal user-mode process. Process ID 1 is usually the init process primarily responsible for starting and shutting down the system.

Originally, process ID 1 was not specifically reserved for init by any technical measures: it simply had this ID as a natural consequence of being the first process invoked by the kernel. More recent Unix systems typically have additional kernel components visible as 'processes', in which case PID 1 is actively reserved for the init process to maintain consistency with older systems.

Process IDs are usually allocated on a sequential basis, beginning at 0 and rising to a maximum value which varies from system to system. Once this limit is reached, allocation restarts at 300 and again increases. In Mac OS X and HP-UX, allocation restarts at 100. However, for this and subsequent passes any PIDs still assigned to processes are skipped.

Some consider this to be a potential security vulnerability in that it allows information about the system to be extracted, or messages to be covertly passed between processes. As such, implementations that are particularly concerned about security may choose a different method of PID assignment.[1] On some systems, like MPE/iX, the lowest available PID is used, sometimes in an effort to minimize the number of process information kernel pages in memory.

The current process ID is provided by a getpid() system call, or as a variable $$ in shell. The process ID of a parent process is obtainable by a getppid() system call. Under Linux, the maximum process ID is given by the pseudo-file /proc/sys/kernel/pid_max.

The file pid_max, which was introduced with the Linux 2.5 kernel, specifies the value at

which PIDs wrap around (i.e., the value in this file is one greater than the maximum PID). It has a default of 32768, but it can be set to any number up to approximately four million. The maximum number of processes on a system is only limited by the amount of physical memory (i.e., RAM) available.

The PIDs for the processes currently on the system can be found by using the ps command or the pstree command (which shows the process names and PIDs in a tree diagram). The top command also shows the PIDs of currently running processes along with other information about them, but it differs in that it continuously updates the information. The pidof command provides the PID of a program whose name is passed to it as an argument (i.e., input).

The PID is needed in order to terminate a frozen or otherwise misbehaving program with the kill command. This command makes it possible to end a program that cannot otherwise be stopped except by rebooting (i.e., restarting) the system, and it is thus an important element in the stability and robustness of Unix-like operating systems.

Information on current processes is stored in the /proc filesystem. This filesystem consists of kernel data that changes in real time (i.e., sensing and responding to external events nearly simultaneously). It is easy to extract the information contained therein using commands such as cat.

Listing the contents of /proc with the ls command as follows will show numerous directories whose names consist only of numbers:

```
ls /proc | less
```

It is convenient to pipe (i.e., transfer) the output from ls /proc to the less command because such output can be fairly long and less allows it to be read one screenful at a time.

There is a numbered directory in /proc corresponding to each PID currently on the system. Each of the directories contains several standard files containing information about the process. For example, the file cmdline contains the name of the command (along with any options and arguments) that the process was started with, and it can be easily read with the cat or head command. For instance, the cmdline file for PID 1 can be read as follows:

```
cat /proc/1/cmdline
```

On most systems it will be necessary to use the root (i.e., administrative) account, which can be accessed by using the su (i.e., substitute user) command, to read files in the /proc directory.

More extensive information can be found about any PID and the process it represents by reading the PID's status file, which is also found in its directory in /proc. For example, the following would display the contents of the status file for PID 1:

```
cat /proc/1/status
```