kernelspace

unix weapons school

Processes

Process 1[1] is launched by the kernel following initialization. The kernel will panic if `init` cannot be found, or if process 1 ever terminates. Process 1 is the ultimate ancestor of all userspace processes, and orphaned processes are reparented to it.

On Linux, new processes are launched with `clone(2)`. GNU libc since 2.3.3 implements `fork(2)` in terms of `clone(2)`. FreeBSD uses the more traditional `rfork(2)` interface. As of FreeBSD 9.1, `fork(2)` is not implemented in terms of `rfork(2)` "for reasons of backwards compatibility.[2]"

`vfork(2)` is unnecessary with modern COW implementations, and has been deprecated by POSIX.1-2008. Do not use it in new code.

[1] Typically `/sbin/init`; override this with kernel parameter `init=`, i.e. "`init=/bin/sh`".

[2] ? Dubious!

POSIX.1-2001 Credentials

A process's PID and PPID, real UID, real GID, and supplementary group IDs are preserved across an `execve(2)`. Effective and set UIDs/GIDs might be changed. Process groups cannot cross session boundaries.

Name	Description	C	Shell	Other
PID	Process ID	`getpid(2)`	$$	`/proc/$$/stat` (f1)
PPID	Parent PID	`getppid(2)`	$PPID	`/proc/$$/stat` (f4)
PGID	Process group ID	`getpgrp(2)`	N/A	`/proc/$$/stat` (f5)
SID	Session ID	`getsid(2)`	N/A	`/proc/$$/stat` (f6)
UID	User ID	`getuid(2)`	$UID	`/proc/$$/status`
UID	User ID	`getuid(2)`	$UID	`/proc/$$/status`
GID	Group ID	`getgid(2)`	N/A	`/proc/$$/status`
EUID	Effective UID	`geteuid(2)`	N/A	`/proc/$$/status`
EGID	Effective GID	`getegid(2)`	N/A	`/proc/$$/status`
SSUID	Saved set-UID	`getresuid(2)`	N/A	`/proc/$$/status`
SSGID	Saved set-GID	`getresgid(2)`	N/A	`/proc/$$/status`
FSUID	Filesystem UID	N/A	N/A	`/proc/$$/status`
FSGID	Filesystem GID	N/A	N/A	`/proc/$$/status`
SGIDS	Supplementary GIDs	`getgroups(2)`	$GROUPS	fixme

Filesystem UID/GID are Linux-specific. Upon an EUID/EGID change, the kernel changes the FSUID/FSGID to match the new values.

OS X supports per-thread credentials using `pthread_setugid_np(2)` and `pthread_getugid_np(2)`. A `pthread_getcred_np` and `pthread_setcred_np` were introduced on the freebsd-arch mailing list in 2009, but have seen little discussion.

Linux uses per-thread credentials in kernelspace, but NPTL enforces the 1-2001 model.

POSIX.1e Capabilities

The superuser concept is very coarse security. Linux implements[3] fine-grained per-thread *capabilities* from the withdrawn POSIX.1e standard, and a wealth of optional “security models” (see `CONFIG_SECURITY`). FreeBSD 9 introduced *capsicum(4)*, a radically different system.

Linux capabilities as of 3.9

- CAP_AUDIT_CONTROL
- CAP_AUDIT_WRITE
- CAP_BLOCK_SUSPEND
- **CAP_CHOWN**
- **CAP_DAC_OVERRIDE**
- CAP_DAC_READ_SEARCH
- **CAP_FOWNER**
- **CAP_FSETID**
- CAP_IPC_LOCK
- CAP_IPC_OWNER
- **CAP_KILL**
- CAP_LEASE
- CAP_LINUX_IMMUTABLE
- CAP_MAC_ADMIN
- CAP_MAC_OVERRIDE
- CAP_MKNOD
- CAP_NET_ADMIN
- CAP_NET_BIND_SERVICE
- CAP_NET_BROADCAST
- CAP_NET_RAW
- **CAP_SETGID**
- CAP_SETFCAP
- CAP_SETPCAP
- **CAP_SETUID**
- CAP_SYS_ADMIN
- CAP_SYS_BOOT
- CAP_SYS_CHROOT
- CAP_SYS_MODULE
- CAP_SYS_NICE
- CAP_SYS_PACCT
- CAP_SYS_PTRACE
- CAP_SYS_RAWIO
- CAP_SYS_RESOURCE
- CAP_SYS_TIME
- CAP_SYS_TTY_CONFIG
- CAP_SYSLOG
- CAP_WAKE_ALARM

Boldface denotes POSIX.1e. All others are Linux-specific.

[3] When the `CONFIG_SECURITY_CAPABILITIES` option is used during kernel build.

Namespaces

Various identifiers are globally shared; each set forms a namespace. Child processes live in their parents' namespaces by default. On Linux, this behavior can be changed via arguments to the `clone(2)` system call at child creation time. A process can change its own namespaces via `setns(2)`[4]

Name	Description	System(s)	`clone(2)` arg
PID	Process IDs	Linux/FreeBSD	`CLONE_NEWPID`
IPC	Interprocess communication (SYSV+POSIX)	Linux	`CLONE_NEWIPC`
↳SYSV IPC	SYSV IPC (shmem, msgqueues, semaphores)	FreeBSD	N/A
↳POSIX IPC	POSIX IPC (message queues)	FreeBSD	N/A
UID	User and group IDs	Linux	`CLONE_NEWUSER`
Net	Network devices, protocol stacks, firewall rules	Linux	`CLONE_NEWNET`
↳Protocols	Protocol endpoints	FreeBSD	N/A
Mount	Filesystem mount points	Linux/FreeBSD	`CLONE_NEWNS`
Paths	Filesystem paths	FreeBSD	N/A
NFS	NFS file handles	FreeBSD	N/A
UTS	`uname(2)` values (node/domain)	Linux	`CLONE_NEWUTS`
Clocks	System clocks	FreeBSD	N/A
MIB	`sysctl` management information base	FreeBSD	N/A
Jails	FreeBSD `jail(8)`s	FreeBSD	N/A

[4] Added in kernel 3.0. `setns(2)` cannot (as of 3.9.2) change all namespaces, only IPC/UTS/Net.