Process group

In a POSIX-conformant operating system, a **process group** denotes a collection of one or more processes.[1] Among other things, a process group is used to control the distribution of a signal; when a signal is directed to a process group, the signal is delivered to each process that is a member of the group.[2]

Similarly, a **session** denotes a collection of one or more process groups.[3] A process may not create a process group that belongs to another session; furthermore, a process is not permitted to join a process group that is a member of another session—that is, a process is not permitted to migrate from one session to another.

When a process replaces its image with a new image (by calling one of the `exec` functions), the new image is subjected to the same process group (and thus session) membership as the old image.

Applications

The distribution of signals to process groups forms the basis of job control employed by shell programs. The TTY device driver incorporates a notion of a **foreground process group**, to which it sends signals generated by keyboard interrupts, notably SIGINT ("interrupt", Control + C), SIGTSTP ("terminal stop", Control + Z), and SIGQUIT ("quit", Control + \). It also sends the SIGTTIN and SIGTTOU signals to any processes that attempt to read from or write to the terminal and that are *not* in the foreground process group. The shell, in turn, partitions the command pipelines that it creates into process groups, and controls what process group is the foreground process group of its controlling terminal, thus determining what processes (and thus what command pipelines) may perform I/O to and from the terminal at any given time.

When the shell `forks` a new child process for a command pipeline, both the parent shell process and the child process immediately make the child process into the leader of the process group for the command pipeline. In this way, it is ensured that the child is the leader of the process group before either the parent or the child relies on this being the case.

Where a textual user interface is being used on a Unix-like system, sessions are used to implement **login sessions**. A single process, the **session leader**, interacts with the controlling terminal in order to ensure that all programs are terminated when a user "hangs up" the terminal connection. (Where a session leader is absent, the processes in the terminal's foreground process group are expected to handle hangups.)

Where a graphical user interface is being used, the session concept is largely lost, and the kernel's notion of sessions largely ignored. Graphical user interfaces, such as where the X display manager is employed, use a different mechanism for implementing login sessions.

Details

The system call `setsid` is used to create a new session containing a single (new) process group, with the current process as both the session leader and the **process group leader** of that single process group.[4] Process groups are identified by a positive integer, the **process group ID**, which is the process identifier of the process that is (or was) the process group leader. Process groups need not necessarily have leaders, although they always begin with one. Sessions are identified by the process group ID of the session leader. POSIX prohibits the change of the process group ID of a session leader.

The system call `setpgid` is used to set the process group ID of a process, thereby either joining the process to an existing process group, or creating a new process group within the session of the process with the process becoming the process group leader of the newly created group.[5] POSIX prohibits the re-use of a process ID where a process group with that identifier still exists (i.e. where the leader of a process group has exited, but other processes in the group still exist). It thereby guarantees that processes may not accidentally become process group leaders.

The system call `kill` is capable of directing signals either to individual processes or to process groups.[2]

See also

- cgroups
- Windows Object Manager, specifically Job objects and associated resource limits[6]

References

- Single UNIX Specification, Issue 6

1. IEEE and The Open Group (2018). "3. Definitions" (https://pubs.opengroup.org/onlinepubs/9699919799/basedefs/V1_chap03.html#tag_03_296). *The Open Group Base Specifications Issue 7*. § 296. Retrieved 2020-08-30. "A collection of processes that permits the signaling of related processes."
2. `kill (https://www.opengroup.org/onlinepubs/9699919799/functions/kill.html)` – System Interfaces Reference, The Single UNIX Specification, Version 4 from The Open Group
3. IEEE and The Open Group (2018). "3. Definitions" (https://pubs.opengroup.org/onlinepubs/9699919799/basedefs/V1_chap03.html#tag_03_343). *The Open Group Base Specifications Issue 7*. § 343. Retrieved 2020-08-30. "A collection of process groups [...]. Each process group is a member of a session."

4. setsid (https://www.opengroup.org/onlinepubs/9699919799/functions/setsid.html) – System Interfaces Reference, The Single UNIX Specification, Version 4 from The Open Group
5. setpgid (https://www.opengroup.org/onlinepubs/9699919799/functions/setpgid.h Single UNIX Specification, Version 4 from The Open Group
6. Karl-Bridge-Microsoft. "Job Objects - Win32 apps" (https://docs.microsoft.com/en-us/windows/ *docs.microsoft.com*. Retrieved 2022-08-28.

Further reading

- McKusick, Marshall Kirk; Neville-Neil, George V. (2004-08-02). "FreeBSD Process Managemer ww.informit.com/articles/article.aspx?p=366888&seqNum=8). *The Design and Implementation* ww.informit.com/title/0201702452). Addison Wesley. ISBN 0-201-70245-2.
- UNIX Signals and Process Groups (http://www.cs.ucsb.edu/~almeroth/classes/W99.276/assign

The **Single UNIX Specification** (**SUS**) is a standard for computer operating systems, compliance with which is required to qualify for using the "UNIX" trademark. The standard specifies programming interfaces for the C language, a command-line shell, and user commands. The core specifications of the SUS known as *Base Specifications* are developed and maintained by the Austin Group, which is a joint working group of IEEE, ISO/IEC JTC 1/SC 22/WG 15 and The Open Group. If an operating system is submitted to The Open Group for certification, and passes conformance tests, then it is deemed to be compliant with a UNIX standard such as UNIX 98 or UNIX 03.