

Advanced Programming in the UNIX Environment

Week 06, Segment 5: Process Limits and Identifiers

**Department of Computer Science
Stevens Institute of Technology**

Jan Schaumann

`jschauma@stevens.edu`

`https://stevens.netmeister.org/631/`



apue\$

getrlimit(2) / setrlimit(2)

```
#include <sys/resource.h>
```

```
int getrlimit(int resource, struct rlimit *rlp);
```

```
int setrlimit(int resource, const struct rlimit *rlp);
```

Returns: 0 on success; -1 on error

Changing resource limits follows these rules:

- a process may change its *soft limit* to a value less than or equal to its *hard limit*
- any process can lower its *hard limit* greater than or equal to its *soft limit*
- only superuser can raise *hard limits*
- changes are per process only (which is why ulimit must be a shell built-in)

apue\$

Process Identifiers

```
#include <unistd.h>

pid_t getpid(void);
pid_t getppid(void);
```

Process ID's are guaranteed to be unique and identify a particular executing process with a non-negative integer.

Certain processes have fixed, special identifiers. They are:

- *swapper, sched, idle* or *system*, process ID 0 – responsible for scheduling
- *init*, process ID 1 – bootstraps a Unix system, owns orphaned processes
- *pagedaemon*, process ID 2 – responsible for the VM system (some Unix systems)



Terminal — 80×24



apue\$

Process Limits and Identifiers

Certain aspects of a process's execution are restricted via resource limits.

A resource limit is specified as a *soft* limit and a *hard* limit; only the superuser may raise a hard limit.

Resource limits are enforced *per process*.

A process further has (at least) a process ID (PID) and a parent process ID (PPID). More on these process relationships in our next videos.