Processing environment

- Process ID
  - Unique
  - Assigned at creation time

PID_MAX
<linux.threads.h>

getpid(void);

Parent process id

getppid(void);

Processing environment

- Process groups
  - OS creates process group when first child is created

- Each process belongs to a process group
  - Process group id
  - Initial process

    process leader

    GPID same as PID

    getpgid(pid);

    getpgid(0); => itself.

- Used for distributing signals
  - Process leader receives a kill signal
    - Distributed to all members of the group

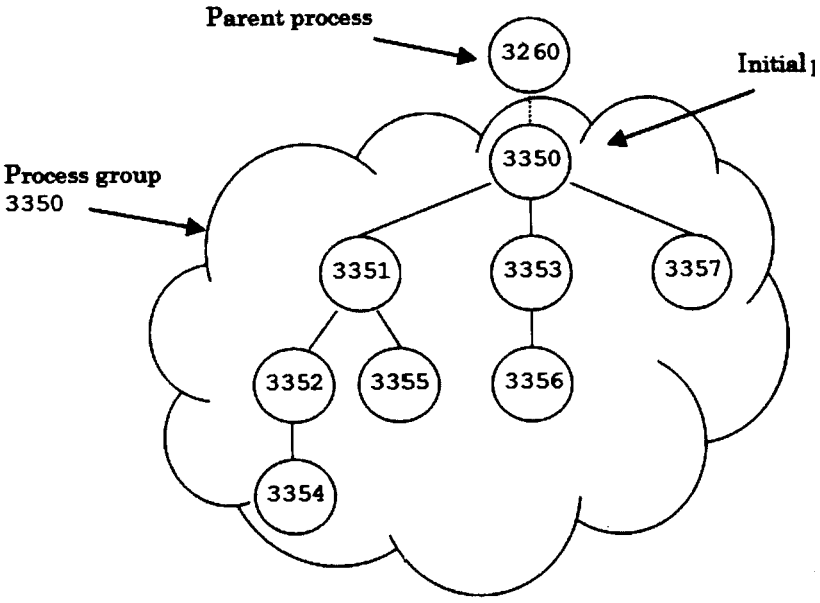


Figure 2.2 Process ID relationships.

```
#define _GNU_SOURCE
#include <iostream>
#include <sys/types.h>
#include <unistd.h>
using namespace std;
int main () {
 cout << "\n\nInitial process \t PID " << getpid()
      << "\t PPID "<< getppid()
      << "\t GID " << getpgid(0)
      << endl <<  getpgid(pid_t(getppid())) << endl;

 for ( int i=0; i<3; i++)
  if ( fork() == 0 )  // generate some processes
   cout << "New process       \t PID " << getpid()
        << "\t PPID "<< getppid()
        << "\t GID  "<< getpgid(0)
        << endl;
 return(0);
}
```

```
Initial process PID 49193   PPID 48874   GID  49193
New process     PID 49194   PPID 1       GID  49193
New process     PID 49195   PPID 1       GID  49193
New process     PID 49196   PPID 1       GID  49193
New process     PID 49197   PPID 1       GID  49193
New process     PID 49198   PPID 49194   GID  49193
New process     PID 49199   PPID 1       GID  49193
New process     PID 49200   PPID 1       GID  49193
```

Why is PPID == 1?

```
...
 for ( int i=0; i<3; i++)
  if ( fork() == 0 )  // generate some processes
   cout << "New process       \t PID " << getpid()
        << "\t PPID "<< getppid()
        << "\t GID  "<< getpgid(0)
        << endl;
sleep (5);
 return(0);
}
```

```
Initial process PID 49447   PPID 49258   GID  49447
New process     PID 49448   PPID 49447   GID  49447
New process     PID 49449   PPID 49447   GID  49447
New process     PID 49450   PPID 49447   GID  49447
New process     PID 49451   PPID 49448   GID  49447
New process     PID 49453   PPID 49449   GID  49447
New process     PID 49454   PPID 49451   GID  49447
New process     PID 49452   PPID 49448   GID  49447
```

Processing environment

- If leader dies
  - adopted by init
  - process group does not change

- Change process group

  setpgid(pid, pgid);

  sets group of pid to pgid

  if pid == 0 => current process

  if pgid == 0 => becomes group leader

Processing environment

- Sessions
  - Collection of related or unrelated processes

    setsid

    getsid

No controlling terminal

Permission

- Permissions
  - rwxrwxrwx
  - Stored a an I-list
    - One unique entry per file
  - Stored in inode table when file is accessed
  - Creation mask

    777 for executables

    666 for text files

    value is EXOR'ED with current mask

- Permissions
  - creation mask
  - Umask => 022
  - Default permissions (file creation)
    - executables 777 EXOR 022 => 755
    - text files 666 EXOR 022 => 644
      - some unix will not set x bit => 655 (rw-r---r--)
    - Directories
      - read: directory can be displayed
      - write: files or links can be created/removed
      - exec: traverse directory

```
-rwxr-xr-x  1 dcanas  dcanas  9172 Sep  3 10:45 a.out
-rw-r--r--  1 dcanas  dcanas   529 Sep  3 10:45 p2.1.cc
```

Processing environment

- Real and effective user/group id
  - Real user/group ID
    - from password file at loggin

```
guest:*:702:710:Guest user:/home/guest:/usr/bin/bash
```

- /etc/group file
- Effective:
  - EUID: effective user id
  - EGID: effective group id
  - Determine additional permissions

Processing environment

- set-user-ID

  SUID

- set-group-ID

  SGID

```
-r-Sr-xr-x   1 root  wheel  35092 Mar 20 19:26 /usr/bin/passwd

-rw-r--r--   1 root  wheel  1932 Aug 22  2005 /etc/passwd
```

Program runs as if were root, effective UID of root

File system information

- Process environment
  - contains file information
  - open files
    - integer file descriptor
      - index to a 1024-entry file descriptor table
    - per process file descriptor table
      - in U area
      - references a system file table

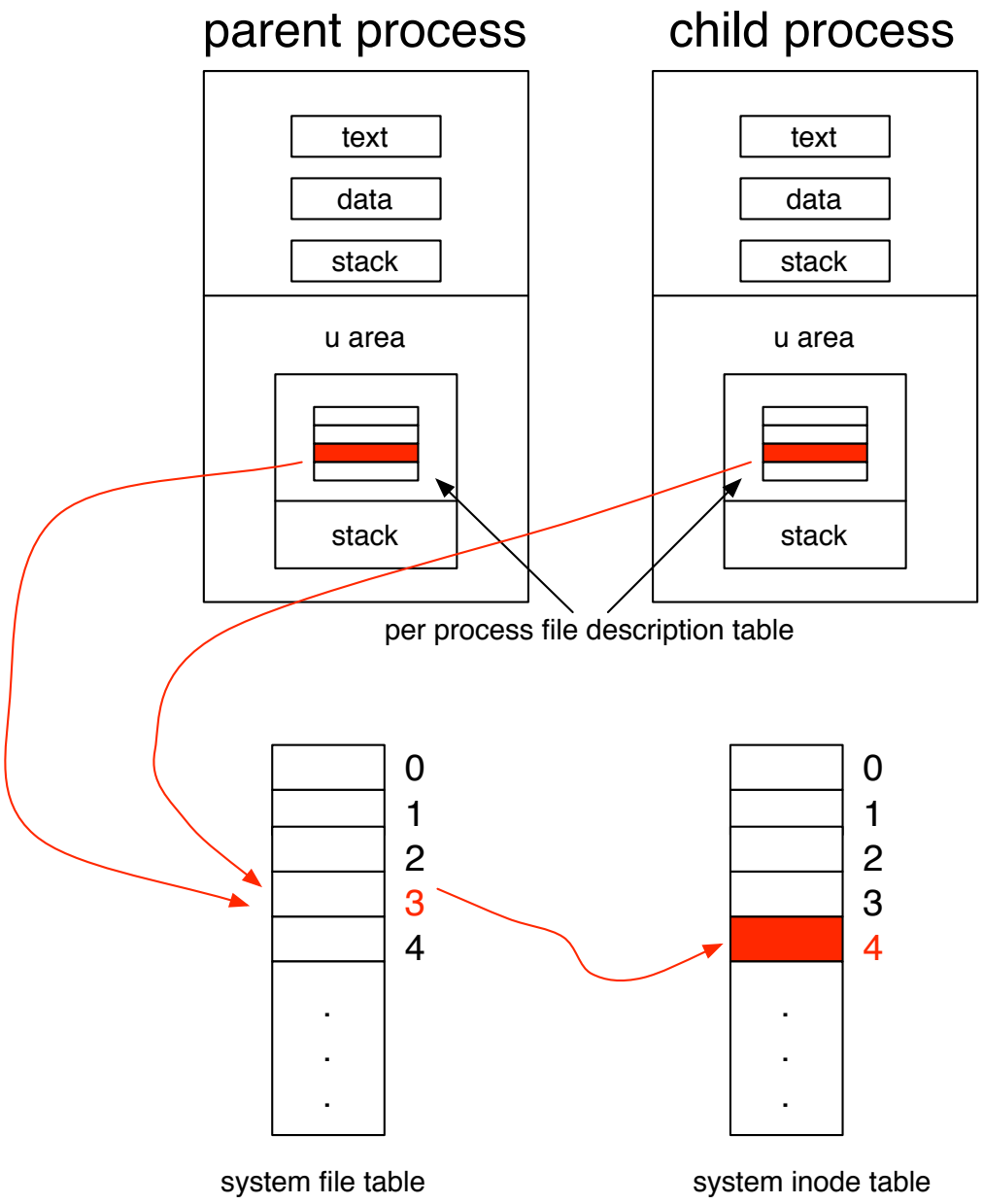
        in kernel space

        maps to a system inode-table

          more complete file information

File system information

Child inherits a copy of parents file descriptor table



HW-1

Exercise 2-4 and 2-5 from textbook

File information

stat system call

```
int stat( const char *file_name, struct stat *buf);
int fstat( int filedes, struct stat *buf);
int lstat( const char *file_name, struct stat *buf);
```

stat: path to file
fstat: file descriptor
lstat: symbolic link

return information in second argument

File system information

Some stat structure entries:

- resident device
- protection
- number of hard links
- user ID of owner
- group ID of owner
- size in bytes
- last access time
- last modify time
- last status change
- actual number of blocks

chmod system call

```
int chmod( const char *path, mode_t mode);
int fchmod(int filedes, mode_t mode);
```

mode: octal number

Other system calls:

getcwd: absolute path of working directory

chdir: change current working directory

fchdir: uses an open file descriptor

ulimit:display and modify system limits

getrlimit/setrlimit

Process resource limits

sysconf (system call)

ulimit

```
ulim  it -Ha    //Hard limits

core file size          (blocks, -c) unlimited
data seg size           (kbytes, -d) unlimited
file size               (blocks, -f) unlimited
max locked memory       (kbytes, -l) unlimited
max memory size         (kbytes, -m) unlimited
open files                      (-n) unlimited
pipe size            (512 bytes, -p) 1
stack size              (kbytes, -s) 65532
cpu time               (seconds, -t) unlimited
max user processes              (-u) 532
virtual memory          (kbytes, -v) unlimited
```

```
ulimit -Sa    //Soft limits

core file size          (blocks, -c) 0
data seg size           (kbytes, -d) unlimited
file size               (blocks, -f) unlimited
max locked memory       (kbytes, -l) unlimited
max memory size         (kbytes, -m) unlimited
open files                      (-n) 256
pipe size            (512 bytes, -p) 1
stack size              (kbytes, -s) 8192
cpu time               (seconds, -t) unlimited
max user processes              (-u) 266
virtual memory          (kbytes, -v) unlimited
```

File system information

- Read sections 2.10 , 2.11 and 2.12

proc

- /proc filesystem
  - Information about
    - Kernel
    - Kernel data structures
    - State of each process
  - Store in memory
  - Use standard open, read to access
  - `procinfo` command

- /proc example

```
#cat cpuinfo
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 15
model           : 4
model name      : Intel(R) Celeron(R) CPU 2.53GHz
stepping        : 1
cpu MHz         : 2527.144
cache size      : 256 KB
fdiv_bug        : no
hlt_bug         : no
f00f_bug        : no
coma_bug        : no
fpu             : yes
fpu_exception   : yes
cpuid level     : 3
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat
pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe pni monitor         ds_cpl est cid
bogomips        : 5046.27
```