Advanced Operating System with UNIX
Unit 3 Processes

Christian Kniep

Internation Center of Applied Technologies Bandung

6. August 2010

Table of content

What is a process?

- A process under UNIX consists of an address space and a set of data structures in the kernel to keep track of that process.

What is a process?

- A process under UNIX consists of an address space and a set of data structures in the kernel to keep track of that process.
- The memory-section contains:
  - the code to execute
  - the process stack

What is a process?

- A process under UNIX consists of an address space and a set of data structures in the kernel to keep track of that process.
- The memory-section contains:
  - the code to execute
  - the process stack
- The kernel keeps track of:
  - adress space map
  - current status of the process
  - execution priority of the process
  - resource usage of the process
  - current signal mask
  - owner of the process

what you might see

- A process has certain attributes, including these:
  PID The unique identifier of a process

what you might see

- A process has certain attributes, including these:
  - PID The unique identifier of a process
  - PPID The parents PID (from whom the process are started)

what you might see

- A process has certain attributes, including these:
  - PID The unique identifier of a process
  - PPID The parents PID (from whom the process are started)
  - UID the UserID of the user who owns the process

what you might see

- A process has certain attributes, including these:
  - PID The unique identifier of a process
  - PPID The parents PID (from whom the process are started)
  - UID the UserID of the user who owns the process
  - GID the GroupID of the owner

what you might see

- A process has certain attributes, including these:
  - PID The unique identifier of a process
  - PPID The parents PID (from whom the process are started)
  - UID the UserID of the user who owns the process
  - GID the GroupID of the owner
  - EUID the EffectiveUserID
  - EGID the EffectiveGroupID

what you might see

- A process has certain attributes, including these:
  - PID The unique identifier of a process
  - PPID The parents PID (from whom the process are started)
  - UID the UserID of the user who owns the process
  - GID the GroupID of the owner
  - EUID the EffectiveUserID
  - EGID the EffectiveGroupID
    - WTF,what is that suppose to mean!?

what you might see

- A process has certain attributes, including these:
  - **PID** The unique identifier of a process
  - **PPID** The parents PID (from whom the process are started)
  - **UID** the UserID of the user who owns the process
  - **GID** the GroupID of the owner
  - **EUID** the EffectiveUserID
  - **EGID** the EffectiveGroupID
    - WTF,what is that suppose to mean!?
  - **Priority** Priority (urgency) the process runs at

The ps-Command

- The command `ps -l`

```
F S UID PID PPID C PRI NI ADDR SZ WCHAN TTY TIME CMD
4 R   0 6843 6842 0 80 0 - 1065 -     pts/0 00:00:00 bash
0 R   0 7552 6843 0 80 0 -  872 -     pts/0 00:00:00 ps
```

The ps-Command

- The command `ps -l`

```
F S UID PID PPID C PRI NI ADDR SZ WCHAN TTY TIME CMD
4 R  0 6843 6842 0 80 0 - 1065 -     pts/0 00:00:00 bash
0 R  0 7552 6843 0 80 0 -  872 -     pts/0 00:00:00 ps
```

- **F** (PROCESS FLAGS)
  - **1** forked but didn't exec
  - **4** used super-user privileges
  - **30** according to the book 'loaded into memory'

The ps-Command

- The command `ps -l`

```
F S UID PID PPID C PRI NI ADDR SZ WCHAN TTY TIME CMD
4 R  0 6843 6842 0 80 0 - 1065 -     pts/0 00:00:00 bash
0 R  0 7552 6843 0 80 0 -  872 -     pts/0 00:00:00 ps
```

- **F** (PROCESS FLAGS)
  - **1** forked but didn't exec
  - **4** used super-user privileges
  - **30** according to the book 'loaded into memory'
- **S** (PROCESS STATE CODES)
  - **D** Uninterruptible sleep (usually IO)
  - **R** Running or runnable (on run queue)
  - **S** Interruptible sleep (waiting for an event to complete)
  - **Z** Defunct (ßombie") process, terminated but not reaped by its parer

The ps-Command

- **PID** The ProcessID

The ps-Command

- **PID** The ProcessID
- **PPID** The ParentProcessID
- (**PGRPID** The ProcessGRouPID)

The ps-Command

- **PID** The ProcessID
- **PPID** The ParentProcessID
- (**PGRPID** The ProcessGRouPID)
- **PRI** The priority of the process

The ps-Command

- **PID** The ProcessID
- **PPID** The ParentProcessID
- (**PGRPID** The ProcessGRouPID)
- **PRI** The priority of the process
- **NI** Nice value; used in priority computation
  - the lower the value of PRI and NI the higher the priority

The ps-Command

- **PID** The ProcessID
- **PPID** The ParentProcessID
- (**PGRPID** The ProcessGRouPID)
- **PRI** The priority of the process
- **NI** Nice value; used in priority computation
  - the lower the value of PRI and NI the higher the priority
  - if a process uses the CPU the PRI-value will raise the nice value

The ps-Command

- **PID** The ProcessID
- **PPID** The ParentProcessID
- (**PGRPID** The ProcessGRouPID)
- **PRI** The priority of the process
- **NI** Nice value; used in priority computation
  - the lower the value of PRI and NI the higher the priority
  - if a process uses the CPU the PRI-value will raise the nice value
- **P** Which processor runs the process

The ps-Command

- **PID** The ProcessID
- **PPID** The ParentProcessID
- (**PGRPID** The ProcessGRouPID)
- **PRI** The priority of the process
- **NI** Nice value; used in priority computation
  - the lower the value of PRI and NI the higher the priority
  - if a process uses the CPU the PRI-value will raise the nice value
- **P** Which processor runs the process
- **SZ (VSZ)** (Virtual) Memory is used

The ps-Command

- **PID** The ProcessID
- **PPID** The ParentProcessID
- (**PGRPID** The ProcessGRouPID)
- **PRI** The priority of the process
- **NI** Nice value; used in priority computation
  - the lower the value of PRI and NI the higher the priority
  - if a process uses the CPU the PRI-value will raise the nice value
- **P** Which processor runs the process
- **SZ (VSZ)** (Virtual) Memory is used
- **TTY** The terminal the process is running at

The ps-Command

- **PID** The ProcessID
- **PPID** The ParentProcessID
- (**PGRPID** The ProcessGRouPID)
- **PRI** The priority of the process
- **NI** Nice value; used in priority computation
  - the lower the value of PRI and NI the higher the priority
  - if a process uses the CPU the PRI-value will raise the nice value
- **P** Which processor runs the process
- **SZ (VSZ)** (Virtual) Memory is used
- **TTY** The terminal the process is running at
- **Time** The cumultative execution time

The ps-Command

- **PID** The ProcessID
- **PPID** The ParentProcessID
- (**PGRPID** The ProcessGRouPID)
- **PRI** The priority of the process
- **NI** Nice value; used in priority computation
  - the lower the value of PRI and NI the higher the priority
  - if a process uses the CPU the PRI-value will raise the nice value
- **P** Which processor runs the process
- **SZ (VSZ)** (Virtual) Memory is used
- **TTY** The terminal the process is running at
- **Time** The cumultative execution time
- **COMD** The executed command

resources

- reward patience
  - those who have used the least will get access served

resources

- reward patience
  - those who have used the least will get access served
  - those who waits for an event will get higher prio (e.g. keyboard)

resources

- reward patience
  - those who have used the least will get access served
  - those who waits for an event will get higher prio (e.g. keyboard)
- The CPU is the most wanted resource. Tied together with the other (such as RAM, Harddrive, Network, etc.) it defines the throughput of an machine

resources

- reward patience
  - those who have used the least will get access served
  - those who waits for an event will get higher prio (e.g. keyboard)
- The CPU is the most wanted resource. Tied together with the other (such as RAM, Harddrive, Network, etc.) it defines the throughput of an machine
- nice

resources

- reward patience
  - those who have used the least will get access served
  - those who waits for an event will get higher prio (e.g. keyboard)
- The CPU is the most wanted resource. Tied together with the other (such as RAM, Harddrive, Network, etc.) it defines the throughput of an machine
- nice
  - the nice-command allows the user to manipulate the internal scheduler

resources

- reward patience
  - those who have used the least will get access served
  - those who waits for an event will get higher prio (e.g. keyboard)
- The CPU is the most wanted resource. Tied together with the other (such as RAM, Harddrive, Network, etc.) it defines the throughput of an machine
- nice
  - the nice-command allows the user to manipulate the internal scheduler
  - only a administrator can set a higher priority (lower value)

resources

- reward patience
  - those who have used the least will get access served
  - those who waits for an event will get higher prio (e.g. keyboard)
- The CPU is the most wanted resource. Tied together with the other (such as RAM, Harddrive, Network, etc.) it defines the throughput of an machine
- nice
  - the nice-command allows the user to manipulate the internal scheduler
  - only a administrator can set a higher priority (lower value)
  - depending on the Unix-flavor the the increments and syntax is different

resources

top have a live updating view on the system processes

resources

top have a live updating view on the system processes

- There are many childs of top with different approaches

htop more information with better formating

ntop network-performance (what process does what)

... and so on

resources

top have a live updating view on the system processes

- There are many childs of top with different approaches

htop more information with better formating

ntop network-performance (what process does what)

... and so on

lsof to have a look what files are opend (maybe have to be installed first)

resources

top have a live updating view on the system processes

- There are many childs of top with different approaches

htop more information with better formating

ntop network-performance (what process does what)

... and so on

lsof to have a look what files are opend (maybe have to be installed first)

fuser what process is using certain files

What is it?

- fork() creates a new process

What is it?

- fork() creates a new process
- the memory allocated to the process will be duplicated

What is it?

- fork() creates a new process
- the memory allocated to the process will be duplicated
- from this line on both processes running the same code

Who was first?

- if a child dies, the parent has to recognize it

Who was first?

- if a child dies, the parent has to recognize it
- if the parent don't, the child will bekomm a 'zombie'

Whats out there?

- **Pipes** Point to point, only one way

Whats out there?

- **Pipes** Point to point, only one way
- **Named Pipes** Looks like a file, thing about it like a spot

Whats out there?

- **Pipes** Point to point, only one way
- **Named Pipes** Looks like a file, thing about it like a spot
- **Message queues** FIFO

Whats out there?

- **Pipes** Point to point, only one way
- **Named Pipes** Looks like a file, thing about it like a spot
- **Message queues** FIFO
- **Semaphores** Guarded line

Whats out there?

- **Pipes** Point to point, only one way
- **Named Pipes** Looks like a file, thing about it like a spot
- **Message queues** FIFO
- **Semaphores** Guarded line
- **Shared Memory** Use the same adresses

Whats out there?

- **Pipes** Point to point, only one way
- **Named Pipes** Looks like a file, thing about it like a spot
- **Message queues** FIFO
- **Semaphores** Guarded line
- **Shared Memory** Use the same adresses
- **Sockets** FIFO in a network

Point to Point with FIFO

- One-Way communication from one entity to another

Point to Point with FIFO

- One-Way communication from one entity to another
- Has an INode but no reference on it, so its **NOT** a file

Point to Point with FIFO

- One-Way communication from one entity to another
- Has an INode but no reference on it, so its **NOT** a file
- with `read` and `write` you could use it

A Pipe With a Name

- Like pipes, but both ways and

A Pipe With a Name

- Like pipes, but both ways and
- they are a file

A Pipe With a Name

- Like pipes, but both ways and
- they are a file

$\Rightarrow$ You have to `open()` it first

a bit organized

a bit organized

- messages have types (nubmer,msg)
- you could read only type-number X

a bit organized

- messages have types (nubmer,msg)
- you could read only type-number X
- could be private (only parent and childs) or public

guard

- its like a queue with a counter

guard

- its like a queue with a counter
- it could seperate enemies

guard

- its like a queue with a counter
- it could seperate enemies
- or unites friends

guard

- its like a queue with a counter
- it could seperate enemies
- or unites friends
- its based on a flag and a queue

on the edge

- if you want be fast, why transfer it?

on the edge

- if you want be fast, why transfer it?
- if you use the same memory-address you don't have to

over the network

- like named pipes not based on files, but on network-ports