

Stony Brook University CSE/ISE 311: Systems Administration

Controlling Processes and the File System

Portions courtesy Ellen Liu

Stony Brook University CSE/ISE 311: Systems Administration

Outline

- Controlling processes (2)
 - Signals and the `kill` command
 - Process monitoring: states, niceness, memory, `ps`, `top`, `uptime`
- The Filesystem
 - Pathnames
 - Mounting and unmounting filesystems
 - File tree organization
 - File types

Stony Brook University CSE/ISE 311: Systems Administration

Signals

- Process-level interrupt requests
- Dozens of them, use "`kill -l`" to list them
- They can be sent
 - among processes as a means to communicate
 - by terminal to kill, interrupt, suspend processes
 - by kernel when encountering e.g., division by zero
 - by kernel to notify. e.g., data arrived on an I/O channel

Stony Brook University CSE/ISE 311: Systems Administration

Upon Receiving a Signal

- A process can "catch it", i.e., designate a signal handler routine to handle it
 - Handler is called. Upon completion, resume (continue) process execution
- A process can also request to block (and then unblock) or ignore signals.
- Otherwise, kernel takes default actions on behalf of the process
 - Generate core dump, or terminate the process

Core dump: a process' memory image, for debugging

Stony Brook University CSE/ISE 311: Systems Administration

Common Signals

#	Name	Description
2	INT	Interrupt (when type ctrl-C)
3	QUIT	Quit
9	KILL	Kill
11	SEGV	Segmentation fault
15	TERM	Software termination
....		

Stony Brook University CSE/ISE 311: Systems Administration

The **kill** Command: Send Signals

- Can send any signals to a process by process owner or the superuser


```
$kill 8021
```

 8021 is the PID
- Default is the SIGTERM, i.e., `kill -TERM`
- SIGTERM may not always terminate a process, `kill -9 8081` sends SIGKILL
 - SIGTERM may be blocked by a process
 - SIGKILL is a signal that can't be blocked by processes

Stony Brook University CSE/ISE 311: Systems Administration

Process States

- **Runnable:** The process can be executed
- **Sleeping:** The process is waiting for some resources
- **Zombie:** terminated but not reaped by its parent
- **Stopped:** The process is suspended (not allowed to execute) or traced

Use the “ps” command to view a process’ state

9-7

Stony Brook University CSE/ISE 311: Systems Administration

Nice and Renice: Scheduling Priority

- Kernel does *process scheduling*: which one do I run next among the Runnable processes?
- Process “niceness” affects the scheduling priority
 - A high nice value means a low priority
 - A low nice value means a high priority
 - In Linux, the range is [-20, 19]
- Owner of a process can increase its nice value but cannot lower it

```
$nice +19 ./myjob10 starts myjob10, and sets it to the lowest priority
```

9-8

Stony Brook University CSE/ISE 311: Systems Administration

The ps Command: Monitor Processes

- Sysadmin’s main tool for monitoring processes
- Shows a process’
 - PID, PPID, UID,
 - control terminal, priority,
 - memory consumption,
 - CPU time used,
 - current status
- **a:** all processes, **x:** even those without terminal, **u:** user oriented output format

9-9

Stony Brook University CSE/ISE 311: Systems Administration

Output of “ps aux”

```
$ ps aux | grep -v guest
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0   600    244 ?        S    14:54   0:03 init [3]
root         2  0.0  0.0     0      0 ?        S    14:54   0:00 [kthreadd]
root         3  0.0  0.0     0      0 ?        SN   14:54   0:00 [ksoftirqd_CPU]
root         4  0.0  0.0     0      0 ?        S    14:54   0:00 [kswapd]
root         5  0.0  0.0     0      0 ?        S    14:54   0:00 [bdf]
root         6  0.0  0.0     0      0 ?        S    14:54   0:00 [kupdate]
root        10  0.0  0.0     0      0 ?        S<   14:54   0:00 [dmccore]
root        11  0.0  0.0     0      0 ?        S    14:54   0:00 [kreiserfsd]
root        57  0.0  0.0     0      0 ?        S    14:55   0:00 [kjournal]
root        80  0.0  0.2  1524   604 ?        Ss   14:55   0:00 /usr/sbin/sshd
root        83  0.0  0.1  1476   460 ?        Ss   14:55   0:00 /usr/sbin/klogd
bin       138  0.0  0.2  1692   616 ?        Ss   14:55   0:00 /sbin/rpc.port
root       671  0.0  0.0     0      0 ?        S    14:55   0:00 [nfsd]
root       672  0.0  0.0     0      0 ?        S    14:55   0:00 [lockd]
root       673  0.0  0.0     0      0 ?        S    14:55   0:00 [rpciod]
root       674  0.0  0.0     0      0 ?        S    14:55   0:00 [nfsd]
root       675  0.0  0.0     0      0 ?        S    14:55   0:00 [nfsd]
root       676  0.0  0.0     0      0 ?        S    14:55   0:00 [nfsd]
root       677  0.0  0.0     0      0 ?        S    14:55   0:00 [nfsd]
```

9-10

Stony Brook University CSE/ISE 311: Systems Administration

Memory Consumed by a Process

- **%MEM:** % of physical (real) memory consumed
- **VSZ:** total amount of virtual memory allocated to the process
- **RSS:** Resident set size (portion of VSZ, i.e., number of pages that are currently in real memory)
- Virtual memory -> physical memory + some disk space
- Managed by pages

9-11

Stony Brook University CSE/ISE 311: Systems Administration

Other Commands

- ps gives only a one-time snapshot of the system
- **top:** provides a regularly updated summary of active processes and their resource consumption
 - By default, every 10 second
- **uptime:** show the up time, the number of users, the load averages (average numbers of runnable processes) over 1, 5, and 15-minute intervals

Read their man pages

9-12

Stony Brook University CSE/ISE 311: Systems Administration

Outline

- Controlling processes (2)
 - Signals and the `kill` command
 - Process monitoring: states, niceness, memory, `ps`, `top`, `uptime`
- The Filesystem
 - Pathnames
 - Mounting and unmounting filesystems
 - File tree organization
 - File types

9-13

Stony Brook University CSE/ISE 311: Systems Administration

The Filesystem

- Represent and organize the system's storage resources, as well as other types of objects – e.g., processes, audio devices, serial ports ...
- Four main components
 - A namespace: name and organize things in a hierarchy
 - An API: system calls to navigate/manipulate objects
 - A security model: scheme to protect/hide/share objects
 - An implementation: software that ties logical model to the hardware

9-14

Stony Brook University CSE/ISE 311: Systems Administration

Pathnames

- The filesystem is a single unified hierarchy that starts at the directory `/`, and continues downward through subdirectories
 - `/`: the root directory
- Pathname: the list of directories that must be traversed to locate a file plus that file's filename
 - Absolute paths: start from root. E.g., `/tmp/foo`
 - Relative paths: start from current directory. E.g., `cse311/A1`
 - Terms pathname, filename, path are interchangeable

9-15

Stony Brook University CSE/ISE 311: Systems Administration

Pathnames (cont'd)

- Filesystem can be arbitrarily deep
- Each pathname must be ≤ 255 characters
 - For longer ones, `cd` to an intermediate directory first, then use a relative pathname
- Filenames
 - Must not contain slash `/` character
 - Spaces are permitted, though not recommended. E.g., `$less "My excellent file.txt"`

9-16

Stony Brook University CSE/ISE 311: Systems Administration

A Portion of the UNIX File Tree

```

graph TD
    root((root)) --- bin((bin))
    root --- usr1((usr))
    root --- tmp((tmp))
    root --- home((home))
    root --- usr2((usr))
    root --- lib((lib))
    root --- dev((dev))
    root --- etc((etc))
    home --- xyzfg((xyzfg))
    home --- abcM((abcM))
    home --- jkl2w((jkl2w))
    usr2 --- News((News))
    usr2 --- docs((docs))
    usr2 --- junk((junk))
    etc --- passwd((passwd))
  
```

9-17

Stony Brook University CSE/ISE 311: Systems Administration

Mounting A filesystem

- Smaller filesystems – each consists of one directory and its subdirectories and files
- Smaller filesystems are attached to the tree with the “mount” command
 - Mount maps a directory in the tree (called mounting point) to the root of the newly attached filesystem
 - `$mount /dev/sda4 /users` install the filesystem stored on the disk partition `/dev/sda4` under the path `/users`.
 - To see the filesystem content, use `ls /users`

9-18

Stony Brook University CSE/ISE 311: Systems Administration

Unmounting A Filesystem

- Filesystems are detached with the “umount” command
 - E.g., `$umount /users`
 - E.g.2, `$umount /mnt/usb` if to umount a USB key device if it was mounted to `/mnt/usb`
- The filesystem can not be busy, i.e., no open files or processes with current directories located there

9-19

Stony Brook University CSE/ISE 311: Systems Administration

Organization of the File Tree

- Every distribution or flavor has slight difference
- Root filesystem: root directory and a small set of files and subdirectories
 - `/bin`: core OS commands
 - `/boot`: kernel and files needed to load the kernel
 - `/dev`: entries for devices, e.g., disks, printers, ...
 - `/etc`: critical startup and configuration files
 - `/home`: default home directories for users
 - `/tmp`: temporary files

9-20

Stony Brook University CSE/ISE 311: Systems Administration

More Standard Directories

- `/lib`: libraries, and parts of the C compiler
- `/mnt`: temporary mount points for removable media
- `/proc`: information about all running processes
- `/root`: home directory of the superuser
- `/usr/bin`: most commands and executables
- `/usr/include`: header files for C compiler
- `/usr/lib`: more libraries
- `/usr/sbin`: less essential commands for sysadmins
- `/var`: log files, accounting info; change rapidly
- ...

9-21

Stony Brook University CSE/ISE 311: Systems Administration

File Types (7 of them)

• Regular files	-	editors, cp	rm
• Directories	d	mkdir	rmdir
• Character device files	c	mknod	rm
• Block device files	b	mknod	rm
• Local domain sockets	s	socket(2)	rm
• Named pipes (FIFOs)	p	mknod	rm
• Symbolic links	l	ln -s	rm

```
$ls -l
-rw-r----- 1 yliu csstaff 4529 Jul 15 2010 todo
```

9-22

Stony Brook University CSE/ISE 311: Systems Administration

File Types (cont'd)

- Regular files**: a series of bytes. Text, data, executable, libraries, etc.
- Directories**: “.” refers to itself, “..” refers to its parent directory. `$cd ..` go to parent dir
- Device files**: used for hardware, peripherals.
 - Characterized by two numbers: major and minor device numbers. Major device number identifies a device driver. Minor tells the driver the actual unit.
 - E.g., the first serial port `/dev/tty0` has 4,0

9-23

Stony Brook University CSE/ISE 311: Systems Administration

File Types (even more)

- Local domain socket**: for connections between processes in local host
 - A filesystem object, not a network port
 - Also called UNIX domain socket
- Named pipes**: similar to above. Both for IPC (inter-process communication)
- Symbolic links**: also called “soft links”

9-24