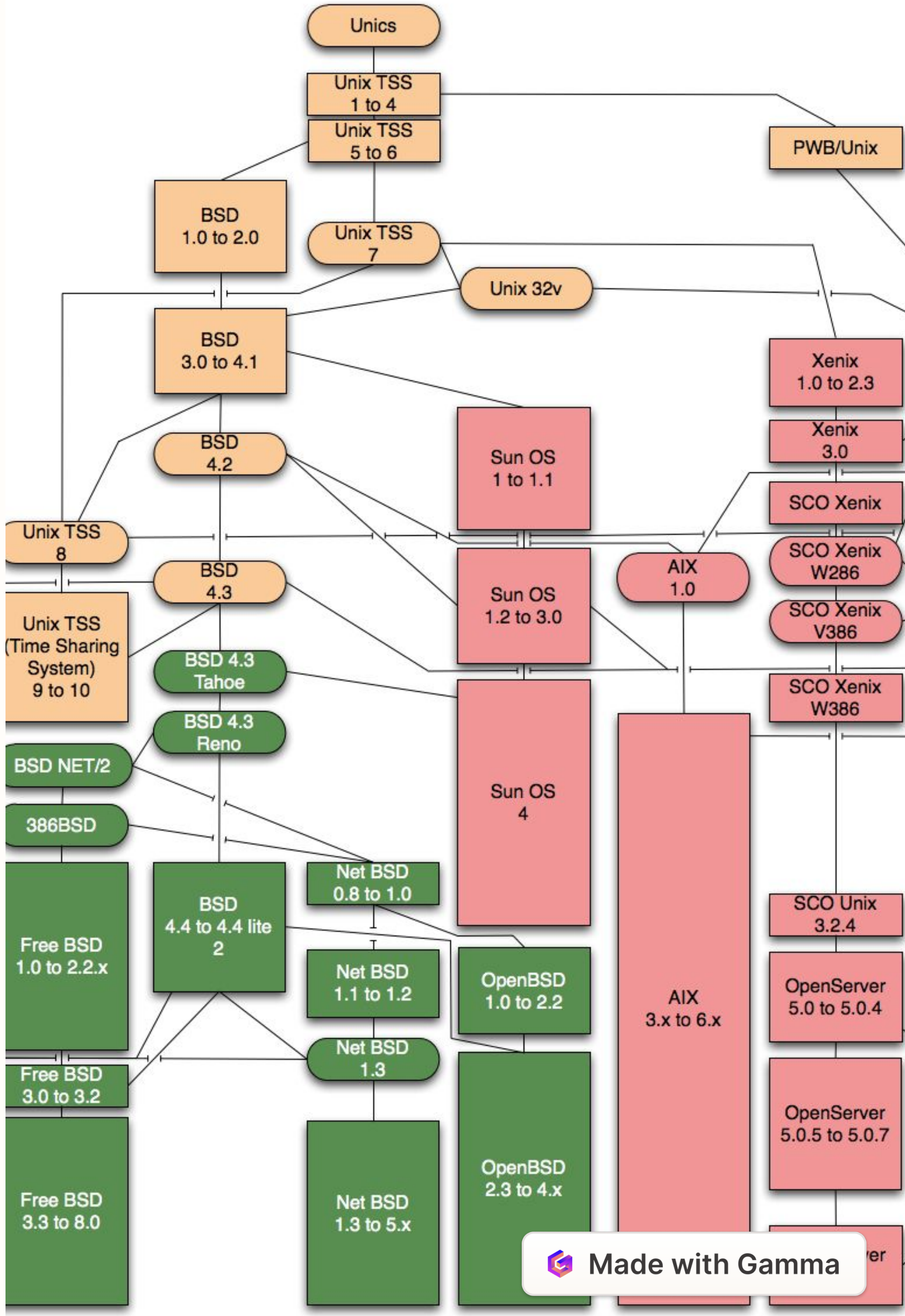
Managing Permissions and Processes on Unix-like Operating Systems

In this presentation, we will discuss some of the most commonly used commands to manage permissions and monitor processes on Unix-like operating systems. You will learn how to control access to files and directories, monitor system activity, and manage running processes.



Made with Gamma

chmod Command: Changing Permissions

The `chmod` command is a powerful tool for controlling who can access files and directories. Here are some of the most common usages:

For Owner

To give read permission: `chmod u+r filename`

To give write permission: `chmod u+w filename`

To give execute permission: `chmod u+x filename`

For Group

To give read permission: `chmod g+r filename`

To give write permission: `chmod g+w filename`

To give execute permission: `chmod g+x filename`

For Others

To give read permission: `chmod o+r filename`

To give write permission: `chmod o+w filename`

To give execute permission: `chmod o+x filename`

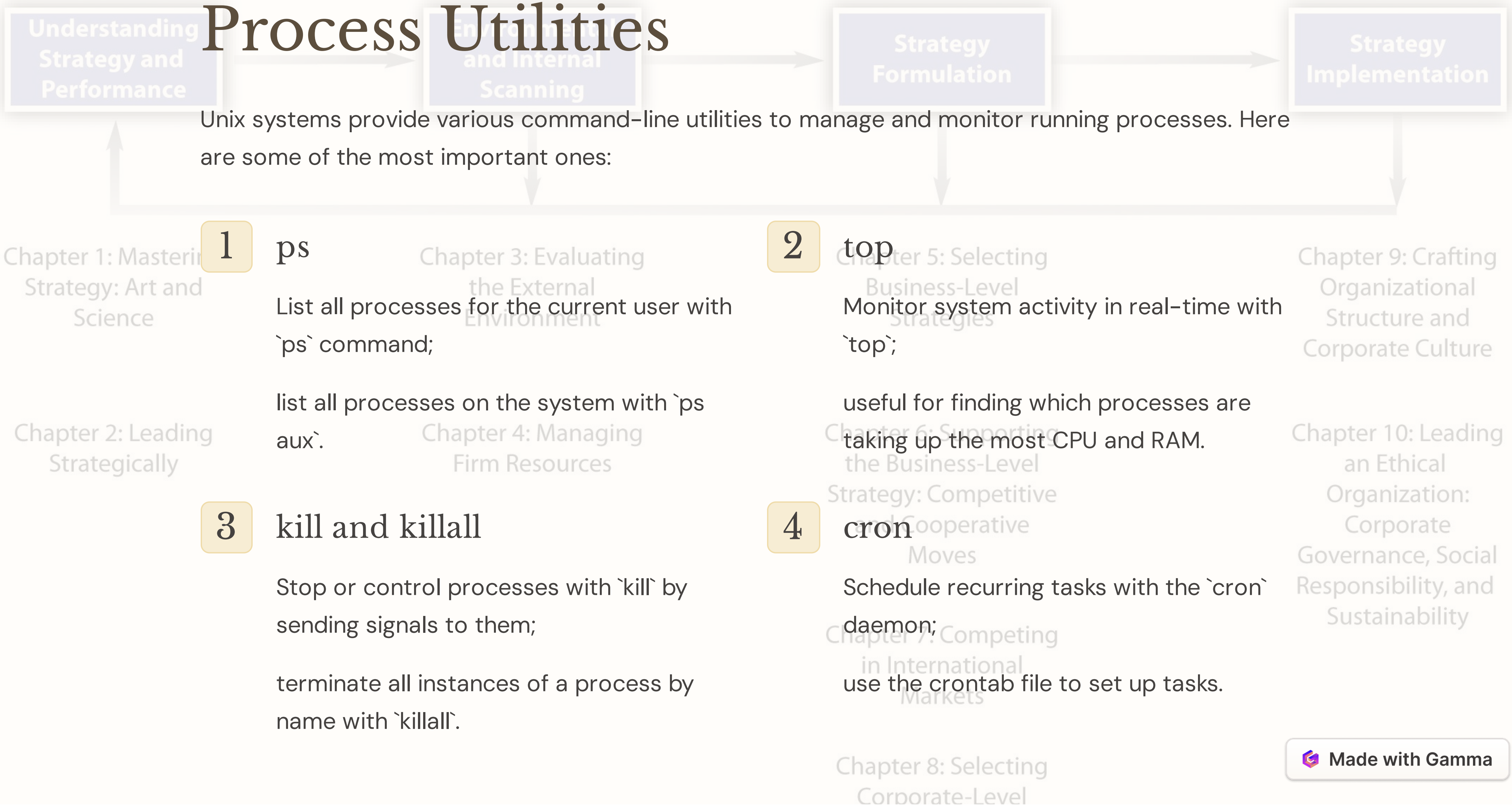
Combining Permissions

To give read, write, and execute permission to owner: `chmod u+rwx filename`

To give read permission to everyone: `chmod a+r filename`

Process Utilities

Unix systems provide various command-line utilities to manage and monitor running processes. Here are some of the most important ones:


Understanding Strategy and Performance
Environmental and Internal Scanning
Strategy Formulation
Strategy Implementation
Chapter 1: Mastering Strategy: Art and Science
Chapter 2: Leading Strategically
Chapter 3: Evaluating the External Environment
Chapter 4: Managing Firm Resources
Chapter 5: Selecting Business-Level Strategies
Chapter 6: Supporting the Business-Level Strategy: Competitive and Cooperative Moves
Chapter 7: Competing in International Markets
Chapter 8: Selecting Corporate-Level
Chapter 9: Crafting Organizational Structure and Corporate Culture
Chapter 10: Leading an Ethical Organization: Corporate Governance, Social Responsibility, and Sustainability

1 ps

List all processes for the current user with `ps` command;

list all processes on the system with `ps aux`.

2 top

Monitor system activity in real-time with `top`;

useful for finding which processes are taking up the most CPU and RAM.

3 kill and killall

Stop or control processes with `kill` by sending signals to them;

terminate all instances of a process by name with `killall`.

4 cron

Schedule recurring tasks with the `cron` daemon;

use the crontab file to set up tasks.

renice and nice

Running multiple processes on a Unix system can lead to competition for components such as CPU and memory. The `renice` and `nice` commands help you prioritize running processes.

1

renice

Increases or decreases the priority of a running process. Useful when you want to give more CPU time to a resource-heavy process.

```
renice -n -5 -p PID
```

2

nice

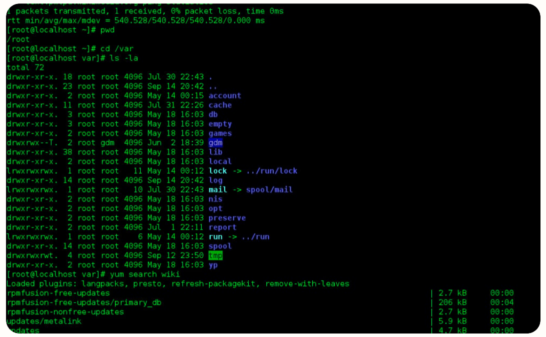
Starts a new process with an adjusted priority. Useful when you want to limit the resources a new process is allowed to consume.

```
nice -n 10 command
```

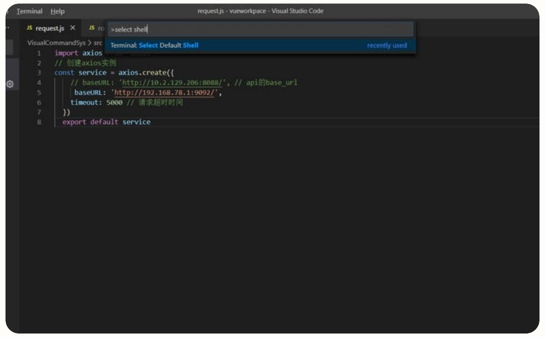
htop and pkill

`htop` is an interactive version of the `top` command, that provides a real-time view of cpu usage, memory usage and more. It enables interactive process management. `pkill` allows you to terminate running processes based on their name.

```
ED                                                    2% [==-
4/repodata/1969b45dd3282b118106f7a50cbe6c8260c388bc49e7d92ce5ac38f7f9f755d4-updateinfo.xml.bz2: [Errno 14] HTTP Error 404 - Not Found

ED                                                    69% [=====================================================-
podata/1969b45dd3282b118106f7a50cbe6c8260c388bc49e7d92ce5ac38f7f9f755d4-updateinfo.xml.bz2: [Errno 14] HTTP Error 404 - Not Found==-

el7 will be installed
                                               https://arkit.co.in
================================================================================================
               Arch                                Version                              Repository
================================================================================================
               x86_64                              2.0.1-1.el7                          epel
================================================================================================

um.
em(s), 'yum check' output follows:
ate with epel-release-7-2.noarch
_64
_64
```



```
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 540.528/540.528/540.528/0.000 ms
[root@localhost ~]# pwd
/root
[root@localhost ~]# cd /var
[root@localhost var]# ls -la
total 72
drwxr-xr-x. 18 root root 4096 Jul 30 22:43 .
drwxr-xr-x. 23 root root 4096 Sep 14 20:42 ..
drwxr-xr-x.  2 root root 4096 May 14 00:15 account
drwxr-xr-x. 11 root root 4096 Jul 31 22:26 cache
drwxr-xr-x.  3 root root 4096 May 18 16:03 db
drwxr-xr-x.  3 root root 4096 May 18 16:03 empty
drwxr-xr-x.  2 root root 4096 May 18 16:03 games
drwxrwx--T.  2 root gdm  4096 Jun  2 18:39 gdm
drwxr-xr-x. 38 root root 4096 May 18 16:03 lib
drwxr-xr-x.  2 root root 4096 May 18 16:03 local
lrwxrwxrwx.  1 root root   11 May 14 00:12 lock -> ../run/lock
drwxr-xr-x. 14 root root 4096 Sep 14 20:42 log
lrwxrwxrwx.  1 root root   10 Jul 30 22:43 mail -> spool/mail
drwxr-xr-x.  2 root root 4096 May 18 16:03 nis
drwxr-xr-x.  2 root root 4096 May 18 16:03 opt
drwxr-xr-x.  2 root root 4096 May 18 16:03 preserve
drwxr-xr-x.  2 root root 4096 Jul  1 22:11 report
lrwxrwxrwx.  1 root root    6 May 14 00:12 run -> ../run
drwxr-xr-x. 14 root root 4096 May 18 16:03 spool
drwxrwxrwt.  4 root root 4096 Sep 12 23:50 tmp
drwxr-xr-x.  2 root root 4096 May 18 16:03 yp
[root@localhost var]# yum search wiki
Loaded plugins: langpacks, presto, refresh-packagekit, remove-with-leaves
rpmfusion-free-updates                                          | 2.7 kB     00:00
rpmfusion-free-updates/primary_db                               | 206 kB     00:04
rpmfusion-nonfree-updates                                       | 2.7 kB     00:00
updates/metalink                                                | 5.9 kB     00:00
updates                                                         | 4.7 kB     00:00
```


Terminal
Help
request.js - vueworkpace - Visual Studio Code
request.js
>select shell
Terminal: Select Default Shell
recently used
VisualCommandSys > src

```
import axios
// 创建axios实例
const service = axios.create({
    // baseURL: 'http://10.2.129.206:8088/', // api的base_url
    baseURL: 'http://192.168.78.1:9092/',
    timeout: 5000 // 请求超时时间
  })
  export default service
```

htop

A better interface for managing processes interactively, this command lets you view all processes running on your system and offers some powerful search and filtering tools as well.

pkill

Instead of using PIDs of the running process, `pkill` allows you to target one or many processes by their name.

Terminating a Process

Using the `kill` command or `htop` or `pkill` you can terminate the processes by their PIDs or name.

at, batch and cron

Sometimes you need to schedule a time for the computer to run tasks. At other times, you need to define when a computer should avoid certain processes. Cron is used to manage predefined jobs that run on a predefined schedule. At and Batch are specifically used to run jobs at a single scheduled times or when the system is idle.

at

In Unix-like systems, `at` allows you to specify a time to run a command or script at. This is useful when you want to perform certain tasks only once at a specified time.

Example:

```
echo "command" | at 2:30pm
```

batch

The `batch` command runs jobs when the system is less loaded. It checks the system load average and runs jobs only if the load average falls below a threshold, which is set by the system's administrator.

Example:

```
batch < job_file
```

cron

`cron` is a time-based job scheduler in Unix-like operating systems. It allows you to schedule jobs to run automatically, such as running backups nightly, performing system maintenance tasks, and others.

Example:

```
0 0 * /path/to/script.sh
```

df and du: Checking Disk Space Usage

Finally, let's take a look at two commands - `df` and `du` - that you can use to check the disk space usage on your system. Both commands are useful to see how much disk space is currently being used by files and directories on your system.

1

df

`df` stands for "disk free" and is used to report on the amount of free and used disk space on your system.

Example: `df -h` shows the amount of available disk space in a human-readable format.

2

du

`du` stands for "disk usage" and is used to estimate the space occupied by a file or a directory.

Example: `du -sh directoryname` shows the summarized size of the directory in a human-readable format.