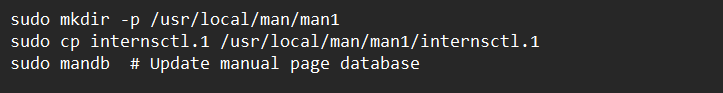
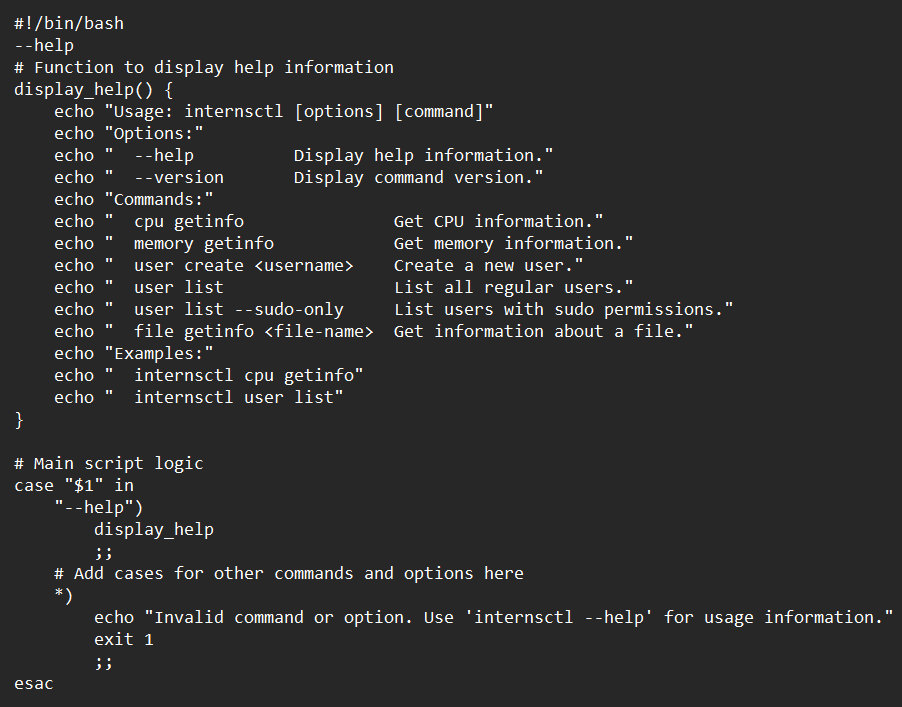
**SECTION – A**

**Installation of Manual**



This section is for installing the manual page for **internsctl**. It creates a directory, copies the manual page (**internsctl.1**), and then updates the manual page database using **mandb**.

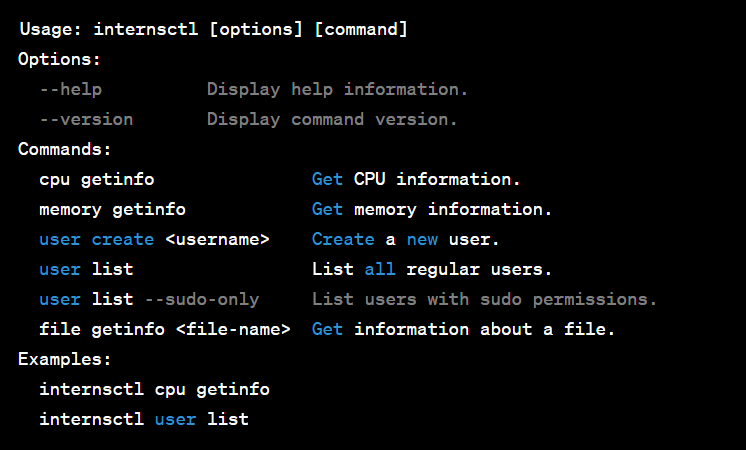
### Bash Script



### Expected Output

1. **Installation:**
   * The installation steps copy the manual page for **internsctl** to the appropriate directory and update the manual page database.
2. **Bash Script:**
   * When you run **internsctl --help**, it should display the help information defined in the **display\_help** function.

Example Output:



The **.TH** header specifies the name, section, date, and version of the manual. The **.SH** (Section Header) and **.TP** (Tagged Paragraph) sections organize the content nicely.

Here's how the manual page breaks down:

### INTERNSCTL Manual

#### **NAME**

**internsctl** - custom Linux command

#### **SYNOPSIS**

**internsctl** [options] [command]

#### **DESCRIPTION**

This command is designed to perform various operations on a Linux system.

#### **OPTIONS**

* **--help:** Display help information.
* **--version:** Display command version.

#### **COMMANDS**

* **cpu getinfo:** Get CPU information.
* **memory getinfo:** Get memory information.
* **user create <username>:** Create a new user.
* **user list:** List all regular users.
* **user list --sudo-only:** List users with sudo permissions.
* **file getinfo <file-name>:** Get information about a file.

#### **EXAMPLES**

* Get CPU information: **internsctl cpu getinfo**
* List all regular users: **internsctl user list**

#### **SEE ALSO**

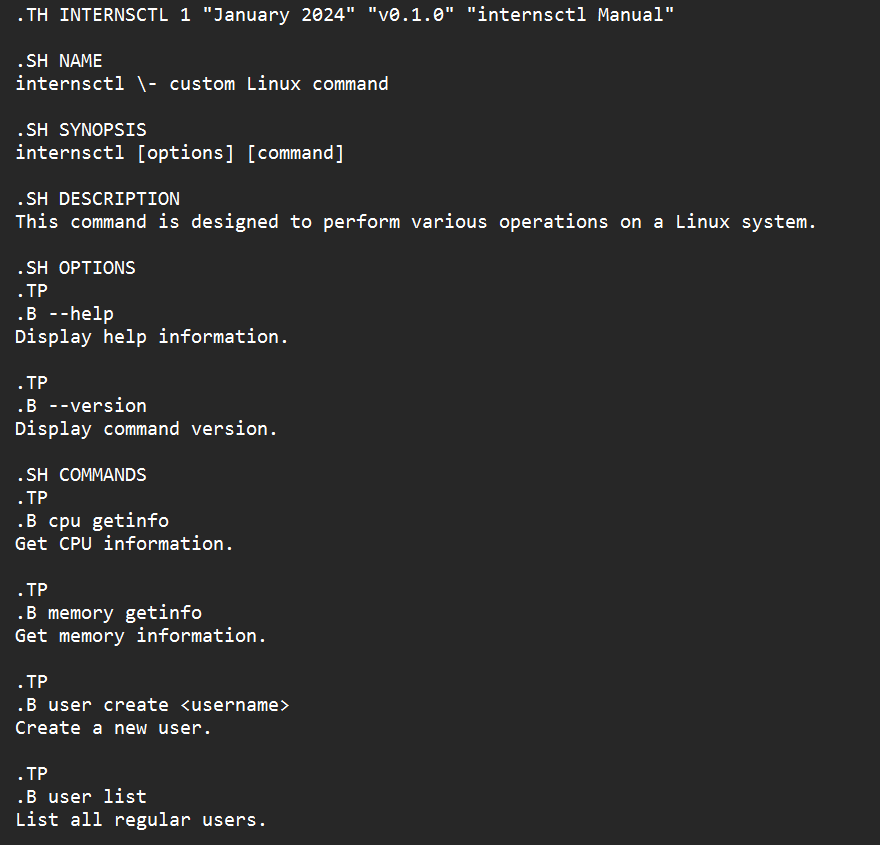
* **lscpu(1)**, **free(1)**, **useradd(8)**, **ls(1)**

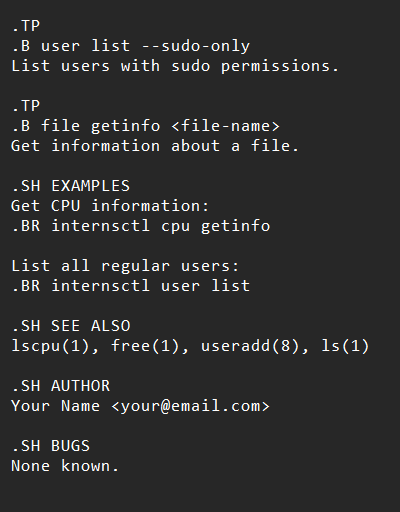
#### **AUTHOR**

Your Name your@email.com

#### **BUGS**

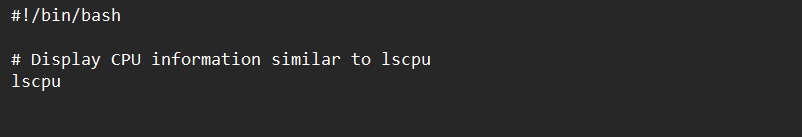
None known.





**SECTION – B**

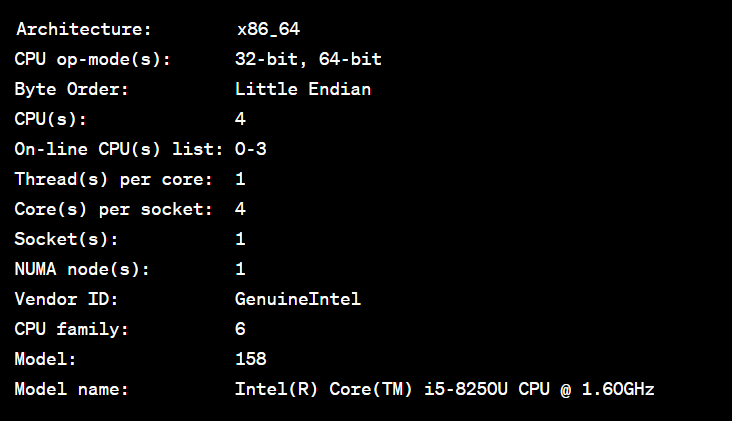
**Part – 1**

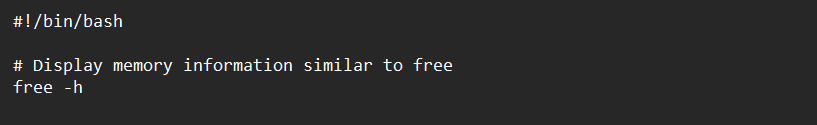
****

* **#!/bin/bash**: This line is called a shebang. It indicates that the script should be executed using the Bash shell.
* **lscpu**: This command is used to display information about the CPU. It provides details like architecture, CPU family, model, number of cores, etc.

### Expected Output:

When you run this script, you'll get output similar to what you'd see when running the **lscpu** command directly in the terminal. For example:

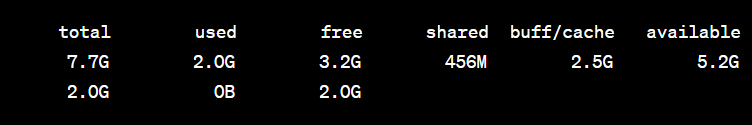
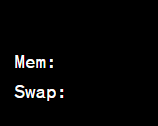
****

****

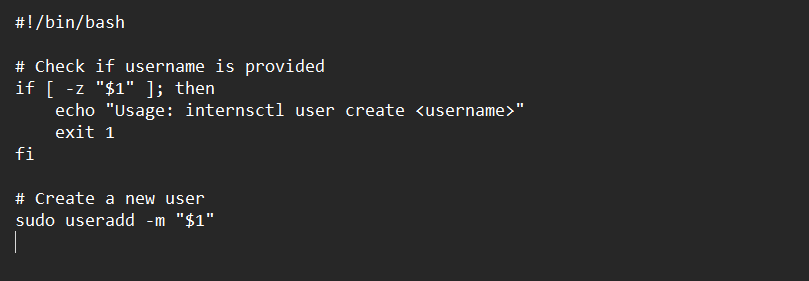
* **#!/bin/bash**: This line is the shebang, indicating that the script should be executed using the Bash shell.
* **free -h**: This command is used to display information about system memory usage and swap space. The **-h** flag makes the output more human-readable.

### Expected Output:

When you run this script, you'll get output similar to what you'd see when running the **free -h** command directly in the terminal. For example:

****

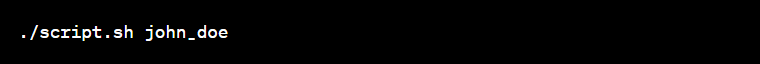
**Part – 2**

****

* **#!/bin/bash**: This line is the shebang, indicating that the script should be executed using the Bash shell.
* The script checks if a username is provided as an argument. If not, it prints a usage message and exits with a status of 1.
* **sudo useradd -m "$1"**: This command creates a new user with the provided username (**$1**) and the **-m** flag ensures that a home directory is created for the user.

### Expected Output:

When you run this script with a username as an argument, it will create a new user. For example:

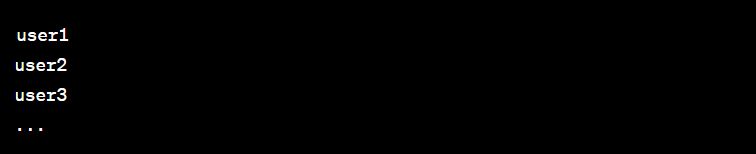
****

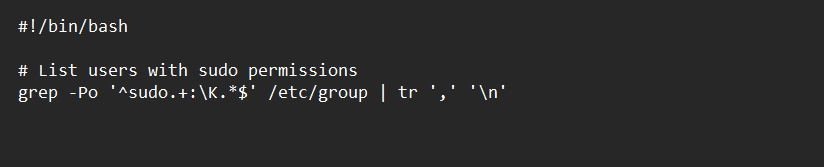
****

* **cut -d: -f1 /etc/passwd**: This command extracts the first field (username) from the **/etc/passwd** file, where fields are delimited by colons (**-d:** specifies the delimiter).

### Expected Output:

When you run this script, it will output a list of all regular user names on the system, each on a new line. For example:

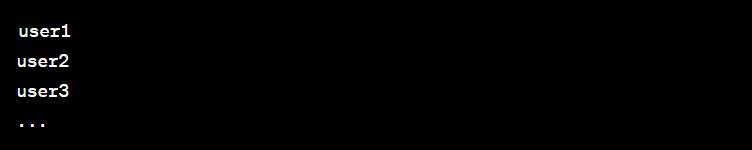
****

****

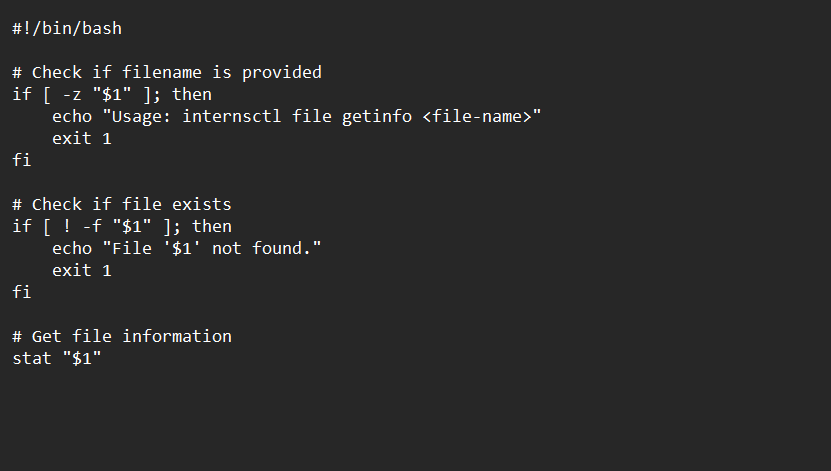
* **grep -Po '^sudo.+:\K.\*$' /etc/group**: This **grep** command extracts the usernames listed after the **sudo** group in the **/etc/group** file.
* **tr ',' '\n'**: This **tr** command replaces commas with newlines, formatting the output to list each username on a new line.

### Expected Output:

When you run this script, it will output a list of users who have sudo permissions, each on a new line. For example:

****

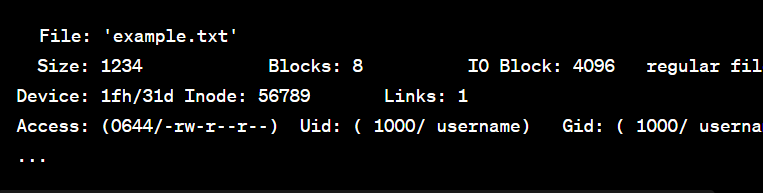
**Part – 3**

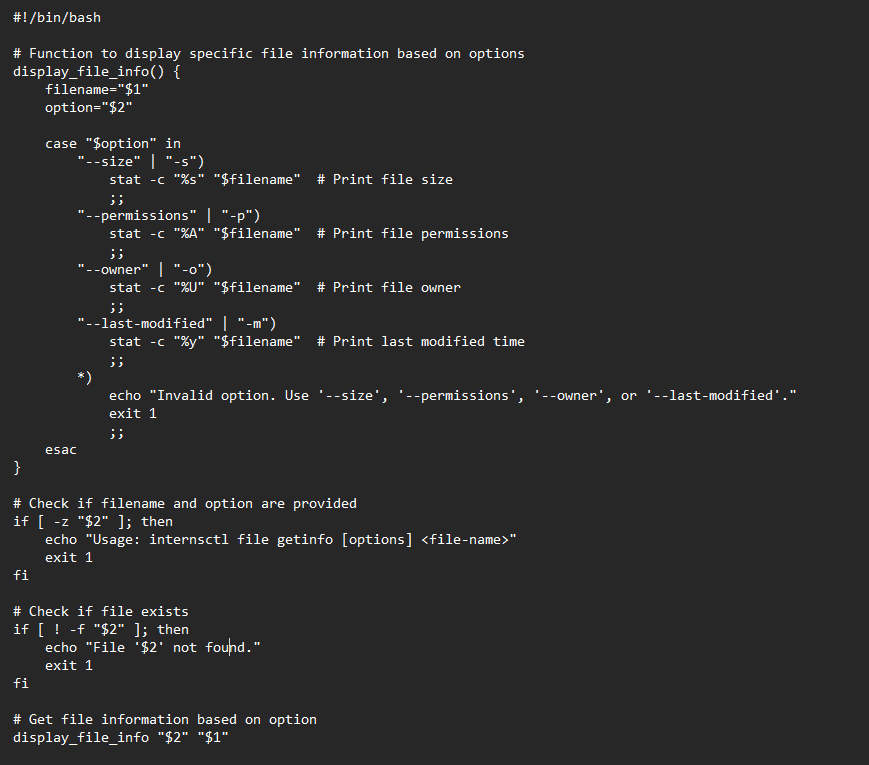
****

* The script checks if a filename is provided as an argument. If not, it prints a usage message and exits with a status of 1.
* It then checks if the specified file exists. If the file does not exist, it prints an error message and exits with a status of 1.
* Finally, if the file exists, it uses the **stat** command to retrieve and display information about the file.

### Expected Output:

When you run this script with a valid filename, it will output information about the specified file using the **stat** command. For example:

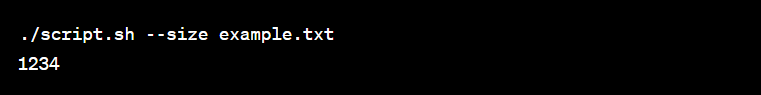
****

****

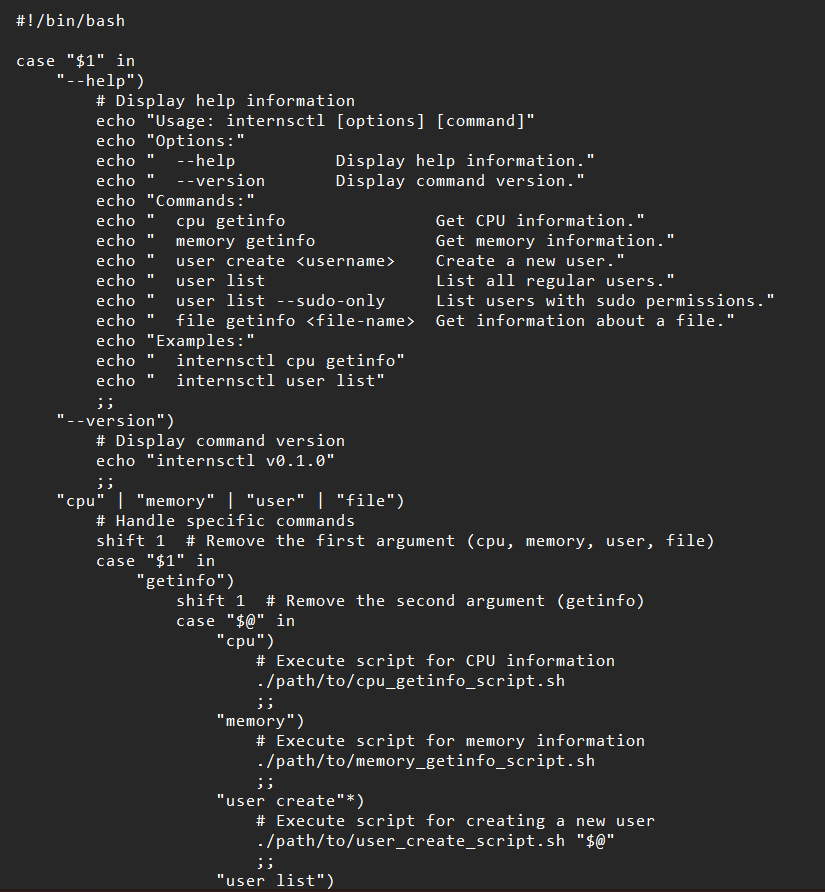
* **display\_file\_info()** is a function that takes a filename and an option as arguments. It uses a **case** statement to determine the option and then retrieves and prints specific information about the file using the **stat** command.
* The script checks if both the filename and option are provided. If not, it prints a usage message and exits with a status of 1.
* It then checks if the specified file exists. If the file does not exist, it prints an error message and exits with a status of 1.
* Finally, it calls the **display\_file\_info** function with the provided filename and option to output the desired file information.

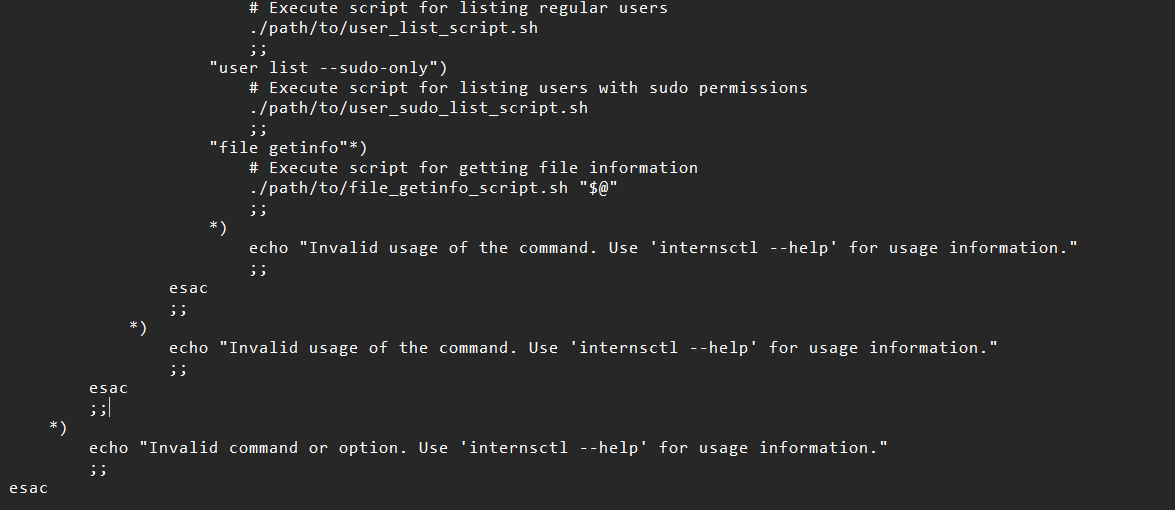
### Expected Output:

When you run this script with a valid filename and option, it will output the specific information about the file based on the provided option. For example:

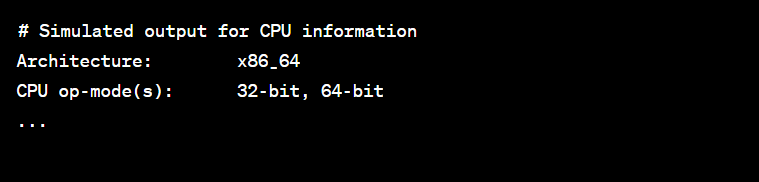
****

**SECTION – C**

****

****

* The script uses a **case** statement to handle different scenarios based on the first argument (**$1**).
* If **--help** is provided, it displays usage information. If **--version** is provided, it displays the command version.
* If a specific command (**cpu**, **memory**, **user**, **file**) is provided, it shifts the arguments to handle the specific subcommands like **getinfo**.
* Depending on the subcommand, it then executes the corresponding script (e.g., **cpu\_getinfo\_script.sh**, **memory\_getinfo\_script.sh**, etc.).
* If an invalid command or option is provided, it displays an error message.

****