

OS-Godz User manual



Help

When passing the **help** command without any arguments the system will return a list of arguments that the user can use in order to gain an understanding of possible commands.

```
> help
The list of commands you can receive help on include:
1. help
2. shutdown
3. time
4. date
5. version
6. pcb
```

The help function can take one of these additional argument which will provide the user with a more detailed explanation of the specified command.

```
> help version
To use version, simply type version and the current version and compilation will display.
o Example: version
```

Version

Entering the **version** command to the system will return the current version the system is operating under along with the timestamp the command was executed at.

```
> version
R1: 12:05:56
```

Time

Entering the **time** command with no parameters will get the current time saved in the system and display it onto the console.

```
> time  
18:08:09
```

Adding an additional parameter after the time command will set the time as long if the input matches the format (HH:MM:SS) where H is the hour, M is the minute, and S is the seconds. Entering the time command with no parameters again will return the current time based off of the time previously set.

```
> time 12:00:00  
Time Set.  
> |
```

An important feature to remember is that the system's clock runs on 24-hour time and is generated in the UTC time zone.

Date

The **date** command without any parameters will get the current date saved within the system and display it on the console.

```
> date  
01/19/23  
> |
```

By adding a parameter to the date command, the user can set the date as long as it follows the (MM/DD/YY) format where M is the month, D is the day, and Y is the year. If the format is entered incorrectly or the date is not within an appropriate range, the system will display a message explaining

where the error occurred and how it can be corrected.

```
> date 01/27/23
Date Set.
> date 01/55/23
o Invalid day. Use 1-31
```

Shutdown

The command used to exit the handler loop is called **shutdown** which will ask the user to confirm their command before executing. In order to confirm this action, retype the shutdown command.

```
> shutdown
o You selected shutdown. Retype shutdown to confirm.
> shutdown
```

Process Control Block (PCB)

After creating a pcb it may become necessary to delete a process. This can be done using the command **pcb delete** followed by the name assigned to the process that is meant to be removed.

```
> pcb delete sample
PCB deleted.
```

An important feature to remember is that a system process cannot be removed by the user and the command will fail if attempted.

The **pcb block** take the name of the process the user wishes to block as its sole parameter.

```
> pcb block sample
PCB blocked.
```

In order to remove this block the user can enter the **pcb unblock** command which takes the name of the block process as it's parameter and unblocks it.

```
> pcb unblock sample
PCB unblocked.
```

There may come a time where the user wishes to suspend a process, in which case they can use the command **pcb suspend** followed by the name of the target process. An important feature to remember is that a system process cannot be suspended by the user.

```
> pcb suspend sample  
PCB suspended.
```

Once the user is ready for a process to resume running once again, they can use the **pcb resume** command followed by the name of the target process in order to remove its suspension.

```
> pcb resume sample  
PCB resumed.
```

The next pcb command will be useful if the user ever desires to make an alteration to the priority of one of the processes created. This is done using the **pcb set priority** command followed by the name of the target process and new priority they wish to set it as. Keep in mind the priority must be within the range of 0 to 9, otherwise the priority cannot be set.

```
> pcb set priority sample 1  
PCB priority set.
```

These last few pcb commands are designed to show the user the current status of the processes on the system. First with the **pcb show 'name'** command which takes the name of the process the user wants to observe and displays its current information including its class, priority, and current state.

```
> pcb show sample  
Name: sample, Class: User, Priority: 3, State: Ready, Suspended: Yes
```

In the event the user wants to see all of the active processes created they can alter the previous command to be **pcb show all**. This command will show all of the processes that currently exist and include their information. A useful facet of this command is that it separates the processes from those which are ready and those which are blocked.

```
> pcb show all
Ready Processes:
Name: sample2, Class: System, Priority: 0, State: Ready, Suspended: No
Name: sample, Class: User, Priority: 3, State: Ready, Suspended: Yes

No blocked processes.
```

If the user wants to filter the processes by their current state there are two commands available that will do so. The **pcb show ready** command will show all of the processes created that are in the ready state.

```
> pcb show ready
Ready Processes:
Name: sample, Class: User, Priority: 1, State: Ready, Suspended: No
```

In contrast the **pcb show blocked** command will show the user all of the processes that are currently in the blocked state.

```
> pcb show blocked
Blocked Processes:
Name: sample2, Class: System, Priority: 0, State: Blocked, Suspended: No
```

Yield

By entering the **yield** command, the user can cause the command handler to yield to the CPU and execute any process within the queue.

Load R3

The **loadr3** command will load in the r3 test processes from processes.h and display them to the user.

This is meant to display how processes are created, loaded, and dispatched concurrently.

```
> loadr3
proc1 dispatched
proc2 dispatched
proc3 dispatched
proc4 dispatched
proc5 dispatched
```

