**potatOS**

(Version 0.4.0)

Technical Support Manual

Table of Contents

[Overview of Program 3](#_Toc433053223)

[Program Structure 4](#_Toc433053224)

[List of Contents in Each File 4](#_Toc433053225)

[Description of Each Function 8](#_Toc433053226)

[Description of Data Structures 22](#_Toc433053227)

[Cross References 24](#_Toc433053228)

[Index 27](#_Toc433053229)

# Overview of Program

A short description of the program including its principle elements

This program, titled potatOS, is an Operating System simulator. In its current state the main goal of the simulator is the design and implementation of an interface as well as some basic functionality. There are currently 22 user commands that the simulator has implemented: version (display version number), date (set or display date), directory (display a list of all of the files in the OS directory), exit (exit the program), suspend (change its state to suspended), resume (change its state to not suspended), setpriority (allow user to change a pcb’s priority value), showpcb (shows all info about a pcb), showall (shows some info about all pcbs), showready (shows some info about all pcbs in the ready state), showblocked (shows some info about all pcbs in the blocked state), sjf (shortest job first), fifo (first in first out), stcf (shortest time to completion first), fpps (fixed priority pre-emptive scheduler), rr (round robin), mlfq (multilevel feedback deque), ls (lottery scheduler), memorymethods(simulates four of the more basic memory allocation methods: first fit, best fit, worst fit, and next fit), and help (displays a list of all of the valid user commands along with a brief description of each). Currently all of the simulator is entirely within a single .cpp file, although this may change with future iterations. The project was written in C++ with the Qt Creator IDE on a computer running Windows 10.

# Program Structure

A description of how the program is set up

## List of Contents in Each File

|  |  |  |
| --- | --- | --- |
| **File Name** | **Function Name** | **Description** |
| main.cpp | InputLoop | A big loop set up to implement the command prompt (i.e. allows the user to repeatedly enter in commands until they enter in a command to quit the program). This function has most of the other functions within it. |
|  | DisplayDate | A simple function that uses some variables passed into it to print the date onto the screen. |
|  | RestoreCurrentDate | Gets the current date from the underlying OS and sets the program’s date variables accordingly. |
|  | GetValidInt | Checks to see if the number that the user has input is valid based on the bounds given. |
|  | DirectoryFIles | Prints a list of all of the file names in this program’s directory. |
|  | DisplayVersion | A simple function that prints the OS’s current version number. |
|  | MaximizeWindow | Gets the computer screen’s resolution and sets the program window to be this size. (note: the sizing portion of this function is currently not working properly) |
|  | ShowHelp | Simply prints a list of each valid command along with a brief description of each. |
|  | ExitProgram | Displays a confirmation message to ensure that the user does want to exit. Then, based on user input, either ends program, returns to command prompt, or displays error message. |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **File Name** | **Function Name** | | | **Description** | |
| main.cpp | AutoTests | | | Performs some automated tests to show that certain functions work. | |
|  |  | | |  | |
|  | AllocatePCB | | | Returns a pointer to a PCB struct. | |
|  | FreePCB | | | Frees the memory of a PCB struct. | |
|  | SetupPCB | | | Sets the values of a PCB struct based on given information. Automatically sets stateOne and stateTwo by default. | |
|  | FindPCB | | | Searches through the two PCB deques to see if there exists a PCB with a user defined name. Return a pointer to it if found, NULL if not. | |
|  | InsertPCB | | | Push a PCB onto a deque based on its state. | |
|  | RemovePCB | | | Remove a PCB from whichever deque it’s in. | |
|  | GetPCBname | | | Ask the user to enter the name of an existing PCB, convert the input to lowercase, and return the string value. | |
|  | ShowPCB | | | Shows information for a single PCB. | |
|  | ShowAll | | | Shows information for all PCBs. | |
|  | ShowReady | | | Shows information for all PCBs in the ready deque. | |
|  | ShowBlocked | | | Shows information for all PCBs in the ready deque. | |
|  | ShortestJobFirst | | | Performs the Shortest Job First process scheduler. | |
|  | FirstInFirstOut | | | Performs the First In First Out process scheduler. | |
|  | ShortestTimeToCompletion | | | Performs the Shortest Time to Completion First process scheduler. | |
| **File Name** | **Function Name** | | | **Description** | |
| main.cpp | FixedPriority | | | Performs the Fixed Priority Pre-Emptive Scheduler. | |
|  | RoundRobin | | | Performs the Round Robin Scheduler. | |
|  | Multilevel | | | Performs the Multilevel Feedback Deque scheduler. | |
|  | Lottery | | | Performs the Lottery Scheduler. | |
|  | ShowMemory | | | Displays a picture of the simulated memory. | |
|  | Coalesce | | | Coalesces the memory by making contiguous smaller blocks of memory into one larger block of memory. | |
|  | Compact | | | Compacts the memory by making everything in memory contiguous starting at address 0. | |
|  | FirstFit | | | Puts a new process into the first place it will fit. | |
|  | NextFit | | | Puts a new process into the next place it will fit. | |
|  | WorstFit | | | Puts a new process into the worst place it will fit (i.e. wherever leaves the biggest memory fragment). | |
|  | BestFit | | | Puts a new process into the best place it will fit (i.e. wherever leaves the smallest memory fragment). | |
|  | MemoryAllocation | | | Performs all of the memory allocation methods. | |
| **File Name** | **Data Structure Name** | | | **Description** | |
| main.cpp | PCB | | | Structure used to implement the PCB object. Values included in a struct PCB include:   1. string processName 2. bool processClass 3. int priority 4. int stateOne 5. int stateTwo 6. int memory 7. struct PCB\* next 8. struct PCB\* previous | |
| **File Name** | **Definition Name/Value** | | | **Description** | |
| main.cpp | APP / true | | | Used for the processClass variable to denote which class the PCB is. | |
|  | SYS / false | | | Used for the processClass variable to denote which class the PCB is. | |
|  |  | | |  | |
| **File Name** | | **Definition Name/Value** | **Description** | |
| main.cpp | | RUNNING / 0 | Used for the stateOne variable to denote whether or not the PCB is running, ready, or blocked. | |
|  | | RUNNING / 0 | Used for the stateOne variable to denote whether or not the PCB is running, ready, or blocked. | |
|  | | READY / 1 | Used for the stateOne variable to denote whether or not the PCB is running, ready, or blocked. | |
|  | | BLOCKED / 2 | Used for the stateOne variable to denote whether or not the PCB is running, ready, or blocked. | |
|  | | NOT\_SUSPENDED / 0 | Used for the stateTwo variable to denote whether or not the PCB is suspended. | |
|  | | SUSPENDED / 1 | Used for the stateTwo variable to denote whether or not the PCB is suspended. | |

# Description of Each Function

An in-depth look at each function and how they work

**Prototype**

void InputLoop();

**Description**

This function is used to create the loop that runs to create a command prompt. Before starting the loop the function displays a welcome message, calls the DisplayVersion function, sets up the date variables to the current date, and calls the MaximizeWindow function. Within the loop a prompt to enter a command is displayed and the function waits for the user to enter in something. The function then sets the user input to lower case and compares their input to a list of acceptable commands (currently through if-else statements but in the future a switch case with enumerated types may be incorporated). Either a valid command will be entered and a corresponding function will be called as a result or an error message will be displayed to let the user know that they have entered an invalid command.

**Prototype**

void DisplayDate(int year, int month, int day);

**Parameters**

|  |  |  |
| --- | --- | --- |
| year | int | Integer value that is used in order to display the current year set. |
| month | int | Integer value that is used in order to display the current month set. |
| day | int | Integer value that is used in order to display the current day set. |

**Description**

This function simply takes in three integer values (year, month, and day) and prints them along with some formatting to display the current date in the system.

**Prototype**

void RestoreCurrentDate(int &year, int &month, int &day);

**Parameters**

|  |  |  |
| --- | --- | --- |
| year | int | Integer value that is used in order to display the current year set. Variable passed by reference in order to manipulate its value within the function. |
| month | int | Integer value that is used in order to display the current month set. Variable passed by reference in order to manipulate its value within the function. |
| day | int | Integer value that is used in order to display the current day set. Variable passed by reference in order to manipulate its value within the function. |

**Description**

The date variables are passed by reference into this function in order to set their values to represent the current date as acquired from the computer. This lets the OS set up the date to be accurate by default as well as allows the user to reset the date to be accurate if they have changed it.

**Prototype**

int GetValidInt(int min, int max);

**Parameters**

|  |  |  |
| --- | --- | --- |
| min | int | Integer value that is used to determine what the minimum acceptable value is for the user to enter. |
| max | int | Integer value that is used to determine what the maximum acceptable value is for the user to enter. |

**Return Value**

Returns an int value equal to what the user had entered if it were acceptable. If the number entered is unacceptable as determined by the min and max variables then an error is displayed telling the user that “You must input a whole number between *min* and *max*” where *min* and *max* are the variables passed into the function.

**Description**

This function simply takes in some parameters to determine the bounds of an acceptable user input and returns the value if it is valid. The function also displays which date value that they are entering a value for (i.e. day, month, or year).

**Prototype**

void DirectoryFiles();

**Description**

This function goes through the OS directory and prints the filename for each file. This function can display two errors: “Not a directory” and “No such file or directory.” These errors are displayed when the file path used to find the program directory is either not a file path at all or if it does not lead to a directory or file. This function currently uses an absolute file path to find the directory so it this function will only work on the computer which it was created on. Therefore this function needs to be altered in order find the file path.

­­­­­­­­­­­­­­­­­­­­**Prototype**

void DisplayVersion();

**Description**

This simple function is just a print statement that displays the program’s version number.

**Prototype**

void MaximizeWindow();

**Description**

This function gets the screen resolution values and stores them into two variables. It is then supposed to set the window size to fullscreen based on these variables but this functionality is currently not working properly.

**Prototype**

void ShowHelp();

**Description**

This function simply uses several print statements to print a list of valid user commands along with a brief description of each.

**Prototype**

void ExitProgram();

**Description**

This function is called when the user wants to exit the program. First the user is asked if they are sure that they want to exit. Based on their input the function will either terminate the program, return back to the command prompt, or display an error to let the user know that they have entered in an unacceptable value and re-display the exit confirmation prompt.

**Prototype**

void AutoTests(int &year, int &month, int &day);

**Parameters**

|  |  |  |
| --- | --- | --- |
| year | int | Integer value that is used in order to display the current year set. Variable passed by reference in order to manipulate its value within the function. |
| month | int | Integer value that is used in order to display the current month set. Variable passed by reference in order to manipulate its value within the function. |
| day | int | Integer value that is used in order to display the current day set. Variable passed by reference in order to manipulate its value within the function. |

**Description**

This function is used to automatically run a short list of tests in order to show that certain functions work properly. The function tests the RestoreCurrentDate, DisplayDate, DirectoryFiles, DisplayVersion, and ShowHelp functions.

**Prototype**

struct PCB\* AllocatePCB();

**Return Value**

Creates a new PCB and returns a pointer to it as long as it does not equal 0. Otherwise it returns NULL.

**Description**

This function simply creates a new struct PCB\* and returns a pointer to it as long as it isn’t 0.

**Prototype**

void FreePCB(struct PCB\* inPCB);

**Parameters**

|  |  |  |
| --- | --- | --- |
| inPCB | struct PCB\* | Pointer to a PCB object. This allows the function to free the memory of the object |

**Description**

This function simply takes a pointer to a PCB object and uses that pointer to free its memory.

**Prototype**

struct PCB\* SetupPCB(string name, int priority, bool classType);

**Parameters**

|  |  |  |
| --- | --- | --- |
| name | string | A string value that will be used to set the name of the PCB object. |
| priority | int | An int value that will be used to set the priority of the PCB object. |
| classType | bool | A Boolean value that will be used to set the class of the PCB object. |

**Return Value**

Returns a pointer to a PCB.

**Description**

Creates a PCB\*, calls the AllocatePCB function, sets the three PCB values that are passed in, sets the two state values by default (stateOne = READY, stateTwo = NOT\_SUSPENDED), and returns a pointer to the PCB.

**Prototype**

struct PCB\* FindPCB(string name, deque<struct PCB\*> ready, deque<struct PCB\*> blocked);

**Parameters**

|  |  |  |
| --- | --- | --- |
| name | string | A string value that will be used to search for a PCB with that name. |
| ready | deque<struct PCB\*> | A deque of all PCBs in the ready state to search through. |
| blocked | deque<struct PCB\*> | A deque of all PCBs in the ready state to search through. |

**Return Value**

Pointer to a PCB if a PCB is found with a matching name, otherwise returns NULL.

**Description**

This function goes through each deque and checks to see if there exists a PCB with the name string passed in. If one is found then a pointer to it is returned, else NULL is returned.

**Prototype**

void InsertPCB(struct PCB\* inPCB, deque<struct PCB\*> \*ready, deque<struct PCB\*> \*blocked);

**Parameters**

|  |  |  |
| --- | --- | --- |
| inPCB | struct PCB\* | A pointer to a PCB object to be inserted into one of the deques. |
| ready | deque<struct PCB\*>\* | A pointer to a deque of PCB objects in the ready state to potentially hold inPCB. |
| blocked | deque<struct PCB\*>\* | A pointer to a deque of PCB objects in the blocked state to potentially hold inPCB. |

**Description**

Takes in a PCB object and pointers to the ready and blocked deques. Then, based on the PCBs stateOne value it will be inserted into the proper deque.

**Prototype**

void RemovePCB(struct PCB\* inPCB, deque<struct PCB\*> \*ready, deque<struct PCB\*> \*blocked);

**Parameters**

|  |  |  |
| --- | --- | --- |
| inPCB | struct PCB\* | A pointer to a PCB object to be inserted into one of the deques. |
| ready | deque<struct PCB\*>\* | A pointer to a deque of PCB objects in the ready state to potentially remove inPCB from. |
| blocked | deque<struct PCB\*>\* | A pointer to a deque of PCB objects in the blocked state to potentially remove inPCB from. |

**Description**

Takes in a PCB object and pointers to the ready and blocked deques. Then, based on the PCBs stateOne value it will be removed from the proper deque and that deque will be updated.

**Prototype**

string GetPCBname();

**Return Value**

Returns a string that the user has entered, but in all lowercase.

**Description**

Asks the user for the name of a PCB, transforms the user input into lowercase, and then returns the string.

**Prototype**

void ShowPCB(deque<struct PCB\*> ready, deque<struct PCB\*> blocked);

**Parameters**

|  |  |  |
| --- | --- | --- |
| ready | deque<struct PCB\*> | A pointer to a deque of PCB objects in the ready state to potentially remove inPCB from. |
| blocked | deque<struct PCB\*> | A pointer to a deque of PCB objects in the blocked state to potentially remove inPCB from. |

**Description**

Shows all information for a single PCB.

**Prototype**

void ShowAll(deque<struct PCB\*> ready, deque<struct PCB\*> blocked);

**Parameters**

|  |  |  |
| --- | --- | --- |
| readyQueue | deque<struct PCB\*> | A pointer to a deque of PCB objects in the ready state to potentially remove inPCB from. |
| blockedQueue | deque<struct PCB\*> | A pointer to a deque of PCB objects in the blocked state to potentially remove inPCB from. |

**Description**

Shows some information for all PCBs.

**Prototype**

void ShowReady(deque<struct PCB\*> ready);

**Parameters**

|  |  |  |
| --- | --- | --- |
| readyQueue | deque<struct PCB\*> | A pointer to a deque of PCB objects in the ready state to potentially remove inPCB from. |

**Description**

Shows some information for all PCBs in the ready deque.

**Prototype**

void ShowBlocked(deque<struct PCB\*> blocked);

**Parameters**

|  |  |  |
| --- | --- | --- |
| blockedQueue | deque<struct PCB\*> | A pointer to a deque of PCB objects in the blocked state to potentially remove inPCB from. |

**Description**

Shows some information for all PCBs in the blocked deque.

**Prototype**

void ShortestJobFirst(deque<struct PCB\*> ready);

**Parameters**

|  |  |  |
| --- | --- | --- |
| readyQueue | deque<struct PCB\*> | A pointer to a deque of PCB objects in the ready state to potentially remove inPCB from. |

**Description**

Performs the Shortest Job First process scheduler.

**Prototype**

void FirstInFirstOut(deque<struct PCB\*> ready);

**Parameters**

|  |  |  |
| --- | --- | --- |
| readyQueue | deque<struct PCB\*> | A pointer to a deque of PCB objects in the ready state to potentially remove inPCB from. |

**Description**

Performs the First In First Out process scheduler.

**Prototype**

void ShortestTimeToCompletion(deque<struct PCB\*> ready);

**Parameters**

|  |  |  |
| --- | --- | --- |
| readyQueue | deque<struct PCB\*> | A pointer to a deque of PCB objects in the ready state to potentially remove inPCB from. |

**Description**

Performs the Shortest Time to Completion First process scheduler.

**Prototype**

void FixedPriority(deque<struct PCB\*> ready);

**Parameters**

|  |  |  |
| --- | --- | --- |
| readyQueue | deque<struct PCB\*> | A pointer to a deque of PCB objects in the ready state to potentially remove inPCB from. |

**Description**

Performs the Fixed Priority Pre-Emptive process scheduler.

**Prototype**

void RoundRobin(deque<struct PCB\*> ready);

**Parameters**

|  |  |  |
| --- | --- | --- |
| readyQueue | deque<struct PCB\*> | A pointer to a deque of PCB objects in the ready state to potentially remove inPCB from. |

**Description**

Performs the Round Robin process scheduler.

**Prototype**

void Multilevel(deque<struct PCB\*> ready);

**Parameters**

|  |  |  |
| --- | --- | --- |
| readyQueue | deque<struct PCB\*> | A pointer to a deque of PCB objects in the ready state to potentially remove inPCB from. |

**Description**

Performs the Multilevel Feedback Queue process scheduler.

**Prototype**

void Lottery(deque<struct PCB\*> ready);

**Parameters**

|  |  |  |
| --- | --- | --- |
| readyQueue | deque<struct PCB\*> | A pointer to a deque of PCB objects in the ready state to potentially remove inPCB from. |

**Description**

Performs the Lottery Scheduler.

**Prototype**

void ShowMemory(struct MemorySpace memory);

**Parameters**

|  |  |  |
| --- | --- | --- |
| memory | struct MemorySpace | A MemorySpace variable which holds what memory is taken and what is free |

**Description**

This function displays a picture representation of the simulated memory space.

**Prototype**

void Coalesce(struct MemorySpace \*memory);

**Parameters**

|  |  |  |
| --- | --- | --- |
| memory | struct MemorySpace\* | A MemorySpace variable pointer which holds what memory is taken and what is free |

**Description**

This function coalesces the simulated memory space by making any contiguous free memory spaces into a single larger memory space.

**Prototype**

void Compact(struct MemorySpace \*memory);

**Parameters**

|  |  |  |
| --- | --- | --- |
| memory | struct MemorySpace\* | A MemorySpace variable pointer which holds what memory is taken and what is free |

**Description**

This function compacts the simulated memory space by making all occupied memory spaces contiguous starting at position 0, thus leaving all free memory contiguous as well.

**Prototype**

bool FirstFit(struct MemorySpace \*memory, struct MemoryBlock \*inBlock);

**Parameters**

|  |  |  |
| --- | --- | --- |
| memory | struct MemorySpace\* | A MemorySpace variable pointer which holds what memory is taken and what is free |
| inBlock | Struct MemoryBlock\* | A MemoryBlock variable pointer which holds information for a new process/memory block. |

**Return Value**

Returns a boolean value that tells whether or not the passed in memory block could be inserted.

**Description**

This function performs the FirstFit memory allocation method by inserting the passed in memory block into the first place it will fit. Returns either true or false, true if it was inserted, false if not.

**Prototype**

bool NextFit(struct MemorySpace \*memory, struct MemoryBlock \*inBlock);

**Parameters**

|  |  |  |
| --- | --- | --- |
| memory | struct MemorySpace\* | A MemorySpace variable pointer which holds what memory is taken and what is free |
| inBlock | Struct MemoryBlock\* | A MemoryBlock variable pointer which holds information for a new process/memory block. |
| position | int | An integer |

**Return Value**

Returns a boolean value that tells whether or not the passed in memory block could be inserted.

**Description**

This function performs the NextFit memory allocation method by inserting the passed in memory block into the next place it will fit based on where the last block was inserted. Returns either true or false, true if it was inserted, false if not.

**Prototype**

bool WorstFit(struct MemorySpace \*memory, struct MemoryBlock \*inBlock);

**Parameters**

|  |  |  |
| --- | --- | --- |
| memory | struct MemorySpace\* | A MemorySpace variable pointer which holds what memory is taken and what is free |
| inBlock | Struct MemoryBlock\* | A MemoryBlock variable pointer which holds information for a new process/memory block. |

**Return Value**

Returns a boolean value that tells whether or not the passed in memory block could be inserted.

**Description**

This function performs the WorstFit memory allocation method by inserting the passed in memory block into the place which will result in the largest amount of excess memory/fragmentation. Returns either true or false, true if it was inserted, false if not.

**Prototype**

bool BestFit(struct MemorySpace \*memory, struct MemoryBlock \*inBlock);

**Parameters**

|  |  |  |
| --- | --- | --- |
| memory | struct MemorySpace\* | A MemorySpace variable pointer which holds what memory is taken and what is free |
| inBlock | Struct MemoryBlock\* | A MemoryBlock variable pointer which holds information for a new process/memory block. |

**Return Value**

Returns a boolean value that tells whether or not the passed in memory block could be inserted.

**Description**

This function performs the BestFit memory allocation method by inserting the passed in memory block into the place which will result in the smallest amount of excess memory/fragmentation. Returns either true or false, true if it was inserted, false if not.

**Prototype**

void MemoryAllocation();

**Description**

This function performs the full memory allocation methods by setting up simulated conditions and displaying how the memory is allocated as each method is run and each block is inserted.

# Description of Data Structures

Use, attributes, and description of the data structures implemented

**Name**

PCB

**Use**

The PCB struct is used to hold all of the information simulate a process control block. The PCB struct is a pivotal piece of the program as a large number of the functions rely on creating and manipulating these objects.

**Attributes**

|  |  |  |
| --- | --- | --- |
| processName | string | A string variable used as an identifier for individual PCBs. The processName will be unique. |
| processClass | bool | A boolean variable used to store which class the PCB is (true = application, false = system). APP and SYS are #define to be equal to true and false, respectively. |
| priority | int | An integer variable used to store the priority of the PCB. This value must be between -127 and +128. |
| stateOne | int | An integer variable used to store what the first state of the PCB is (0 = running, 1 = ready, 2 = blocked). RUNNING, READY, and BLOCKED are #define to 0, 1, and 2, respectively. |
| stateTwo | int | An integer variable used to store what the second state of the PCB is (0 = not suspended, 1 = suspended). NOT\_SUSPENDED and SUSPENDED are #define to equal 0 and 1, respectively. |
| memory | int | An integer variable used to store the amount of memory required by the PCB. |
| next | struct PCB\* | A pointer to the next object in the PCB’s deque. |
| previous | struct PCB\* | A pointer to the previous object in the PCB’s deque. |

**Name**

MemoryBlock

**Use**

The MemoryBlock struct is used to hold the start (min) and end (max) memory positions for a process. It also holds a process’s name to tell which process it is related to and a boolean value to tell whether or not the block is in use.

**Attributes**

|  |  |  |
| --- | --- | --- |
| processName | string | A string variable used as an identifier for individual PCBs. The processName will match a PCB’s process name. |
| min | int | A integer variable used to store the start location of the memory block’s place in memory. |
| max | int | An integer variable used to store the end location of the memory block’s place in memory. |
| free | bool | A boolean variable used to tell whether or not the memory block is in use (i.e. whether or not the process has completed execution). |

**Name**

MemorySpace

**Use**

The MemorySpace struct is used to hold the size of our simulated memory space as well as a vector holding each process that is in memory.

**Attributes**

|  |  |  |
| --- | --- | --- |
| min | int | An integer variable used to store the base location of our memory. |
| max | int | An integer variable used to store the end location of our memory. |
| occupiedMemory | vector<struct MemoryBlock> | A MemoryBlock vector used to store each process that is in memory. |

# Cross References

A representation of what each function calls and is called by

|  |  |  |
| --- | --- | --- |
| **Function Name** | **What Does This Call?** | **What Calls This?** |
| InputLoop | DisplayVersion, RestoreCurrentDate, MaximizeWindow, DisplayVersion, DisplayDate, GetValidInt, DirectoryFiles, ShowHelp, ExitProgram, GetPCBname, FindPCB, InsertPCB, SetupPCB, RemovePCB, FreePCB, ShowPCB, ShowAll, ShowReady, ShowBlocked, ShortestJobFirst, FirstInFirstOut, ShortestTimeToCompletion, FixedPriority, RoundRobin, Multilevel, Lottery, MemoryAllocation | main |
| DisplayDate | [none] | InputLoop, AutoTests |
| RestoreCurrentDate | [none] | InputLoop, AutoTests |
| GetValidInt | [none] | InputLoop |
| DirectoryFiles | [none] | InputLoop, AutoTests |
| DisplayVersion | [none] | InputLoop, AutoTests |
| MaximizeWindow | [none] | InputLoop |
| ShowHelp | [none] | InputLoop, AutoTests |
| ExitProgram | [none] | InputLoop |
| AutoTests | RestoreCurrentDate, DisplayDate, DirectoryFiles, DisplayVersion, ShowHelp | Main |
| AllocatePCB | [none] | SetupPCB |
| **Function Name** | **What Does This Call?** | **What Calls This?** |
| FreePCB | [none] | InputLoop |
| SetupPCB | AllocatePCB | InputLoop |
| FindPCB | [none] | InputLoop |
| InsertPCB | [none] | InputLoop |
| RemovePCB | [none] | InputLoop |
| GetPCBname | [none] | InputLoop |
| ShowPCB | [none] | InputLoop |
| ShowAll | [none] | InputLoop |
| ShowReady | [none] | InputLoop, ShortestJobFirst, FirstInFirstOut, ShortestTimeToCompletion, FixedPriority, RoundRobin, Multilevel, Lottery |
| ShowBlocked | [none] | InputLoop |
| ShortestJobFirst | [none] | InputLoop |
| FirstInFirstOut | [none] | InputLoop |
| ShortestTimeToCompletion | [none] | InputLoop |
| FixedPriority | [none] | InputLoop |
| RoundRobin | [none] | InputLoop |
| Multilevel | [none] | InputLoop |
| Lottery | [none] | InputLoop |

|  |  |  |
| --- | --- | --- |
| **Function Name** | **What Does This Call?** | **What Calls This?** |
| ShowMemory | [none] | MemoryAllocation |
| Coalesce | [none] | MemoryAllocation |
| Compact | [none] | MemoryAllocation |
| FirstFit | [none] | MemoryAllocation |
| NextFit | [none] | MemoryAllocation |
| WorstFit | [none] | MemoryAllocation |
| BestFit | [none] | MemoryAllocation |
| MemoryAllocation | ShowMemory, Coalesce, Compact, FirstFit, NextFit, WorstFit, BestFit | main |

# Index

AllocatePCB, 11, 12, 24, 25

AutoTests, 5, 11, 24

BestFit, 6, 20, 26

Coalesce, 6, 26

Compact, 6, 26

DirectoryFIles, 4

DisplayDate, 4, 8, 11, 24

DisplayVersion, 4, 8, 10, 11, 24

ExitProgram, 4, 11, 24

FindPCB, 5, 13, 18, 19, 20, 21, 24, 25

FirstFit, 6, 19, 26

FirstInFirstOut, 5, 16, 24, 25

FixedPriority, 6, 17, 24, 25

FreePCB, 5, 12, 24, 25

GetPCBname, 5, 14, 24, 25

GetValidInt, 4, 9, 24

IDE, 3

InputLoop, 4, 5, 7, 8, 24

InsertPCB, 5, 13, 24, 25

Lottery, 6, 17, 25

MaximizeWindow, 4, 8, 10, 24

MemoryAllocation, 6, 26

Multilevel, 6, 17, 24, 25

NextFit, 6, 19, 26

RemovePCB, 5, 14, 24, 25

RestoreCurrentDate, 4, 9, 11, 24

RoundRobin, 6, 17, 24, 25

SetupPCB, 5, 12, 24, 25

ShortestJobFirst, 5, 16, 24, 25

ShortestTimeToCompletion, 5, 16, 24, 25

ShowHelp, 4, 10, 11, 24

ShowMemory, 6, 26

WorstFit, 6, 20, 26