# VisualOS

## User Manual

**Manuel Estrada Sainz**

**ranty@debian.org**
**ranty@bigfoot.com**

**VisualOS: User Manual**

by Manuel Estrada Sainz

# Table of Contents

# Chapter 1. Introduction

VisualOS is an operating system simulator with the purpose of allowing easier and faster learning of such a beast's internals. It shows visually the most important mechanisms and allows manipulation the running system without writing a single line of code.

It's structure is quite simple. Initially each simulated subsystem is located in it's own window. The CPU is the center of the system and the originator of all activity requesting services from all others.

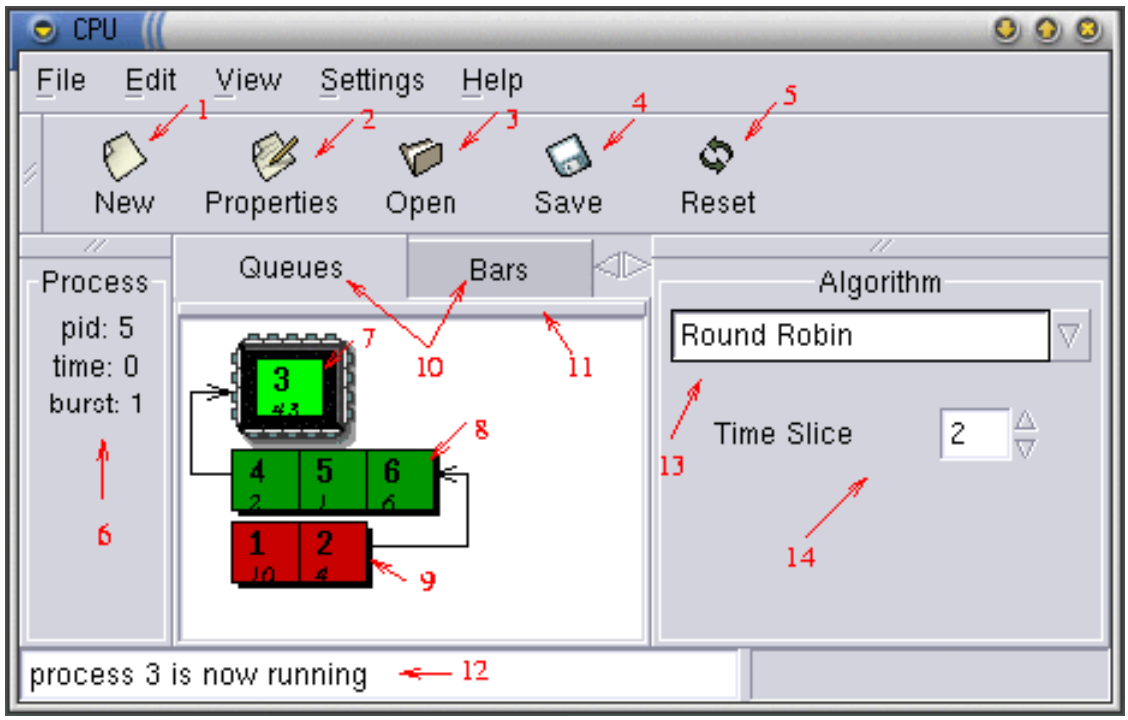
## Common structure

All subsystems have a similar structure and certain peculiarities which will be explained later.

All of them allow the selection of the algorithm to be used and it's parameters, if it has any. They also offer different representations of what is going on inside.

# Chapter 2. The CPU

In the CPU you can find processes which are the originators of all action. It’s elements are described below:



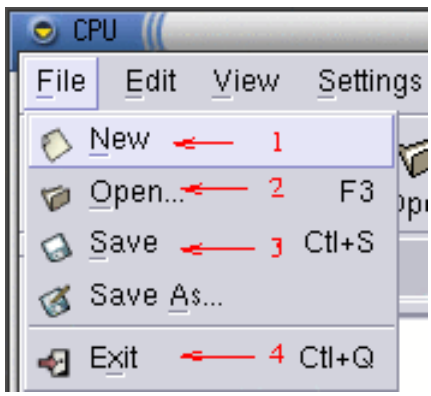

1. Create a new process, it will usually allow selection of it’s characteristics.
2. Modify the characteristics of the currently selected process.
3. Load one of more processes from a file. This file may be created by the program or hand written by the user.
4. Save current session with all it’s processes to a file. All processes will be saved, including already terminated processes and processes not yet inserted in the system.
5. Reset system; returns the system to it’s initial state, ready to start a new session.
6. PID, execution time, and current burst of the currently selected process.
7. Currently running process.
8. "Ready to run" process queue; Multiple queues of this type are allowed depending on the algorithm in use.
9. Blocked process queue, waiting for a disk read or a page fault to complete.
10. All different representations available for the systems.

11. Click on this button to put the subsystem representation in it's own window, which allows multiple simultaneous representations of the same subsystem to be visible at the same time.
12. Messages telling interesting events.
13. The currently selected algorithm's name. Any of the algorithms in the combo may be selected, including the manual "algorithm" which allows the user to make the choices.
14. The currently selected algorithm's properties. In this case it is the time slice that each process is allowed to run at once.

## The menus

Here we will see what the options on each menu mean:

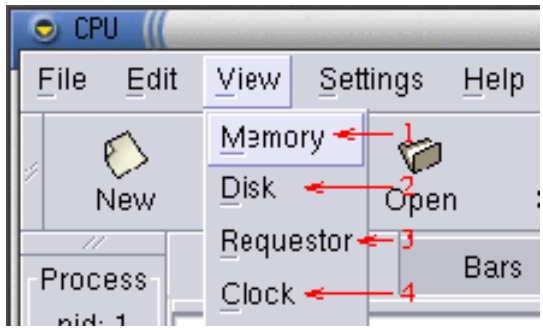
### File




1. Create a new process, it will usually allow selection of it's characteristics.
2. Load one of more processes from a file. This file may be created by the program or hand written by the user.
3. Save current session with all it's processes to a file. All processes will be saved, including already terminated processes and processes not yet inserted in the system.
4. Exit the program closing all subsystems.

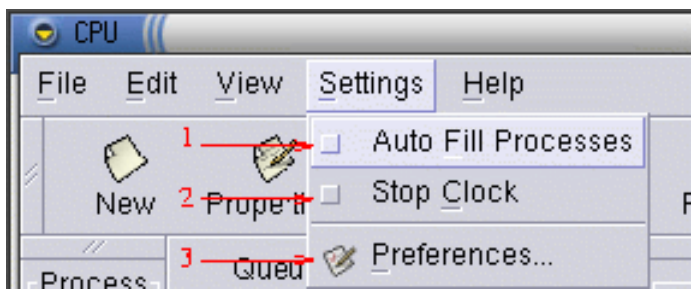
### View

This menu allow showing the main windows from the other subsystems when they have been hidden.




1. Shows the memory subsystem.
2. Shows the disk subsystem.
3. Shows the "Requestor" subsystem (We will explain this one later).
4. Show the clock subsystem.

### Settings

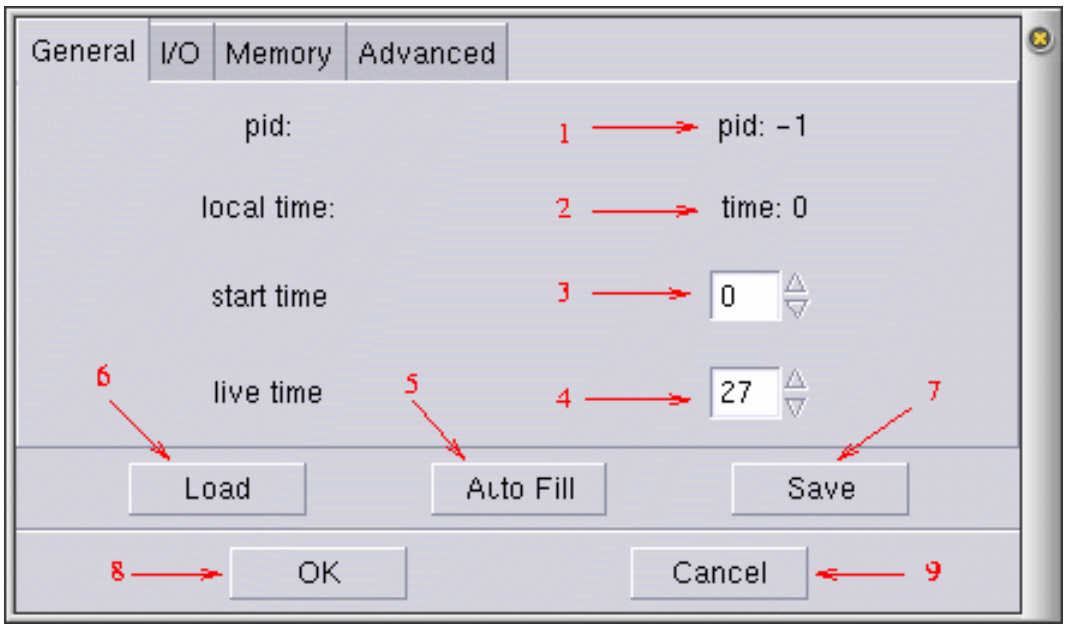



1. When this is checked, the process properties window will not pop up when creating one. Its properties will be filled automatically based on "auto-filling" properties.
2. When this is checked, the processor stops the clock whenever something interesting happens.
3. Show the preferences window, which we will see later.

## Process properties

This window allows the specification of a process' properties, and also loading and saving a single process to file.

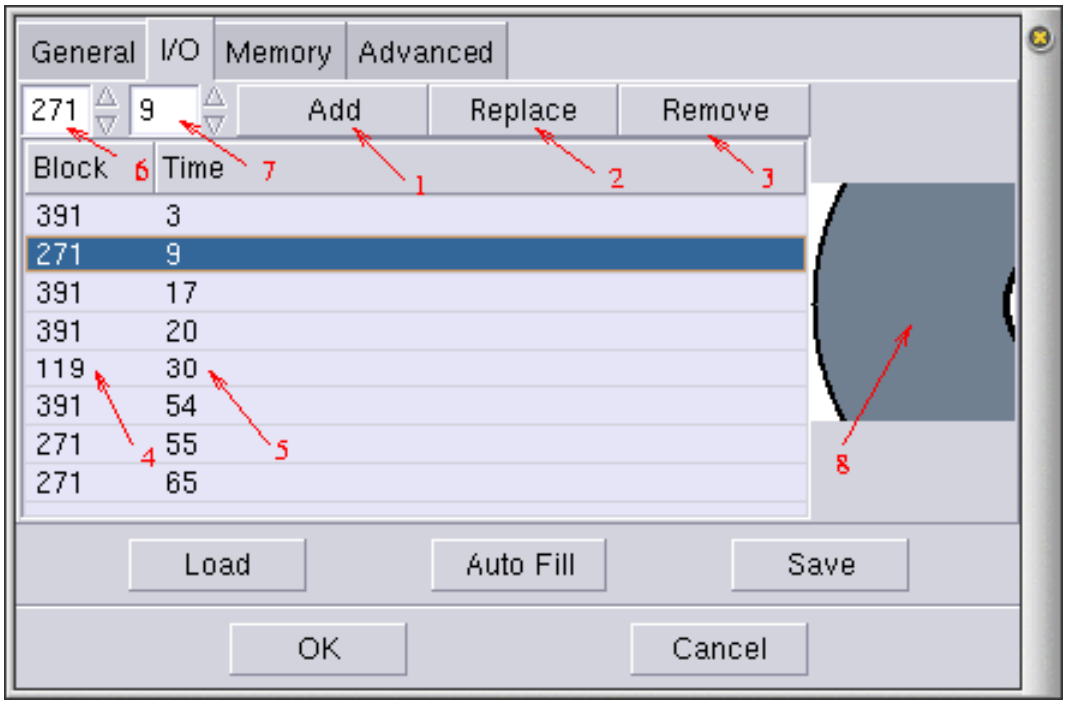
## General




1. Process identifier, -1 if the process has not yet been inserted into the system.
2. Processor time spent by the process.
3. Time at which the process will be inserted into the system.
4. Processor time that the process will spend within all it’s lifetime.
5. Auto-fill current properties page.
6. Load the process from file.
7. Save the process to file.
8. Accept the changes made to the properties.
9. Cancel changes made to the properties and if we are creating a new process, cancel its creating.

## I/O

This are the accesses to secondary memory ("I/O") that the process will do in all it’s lifetime.

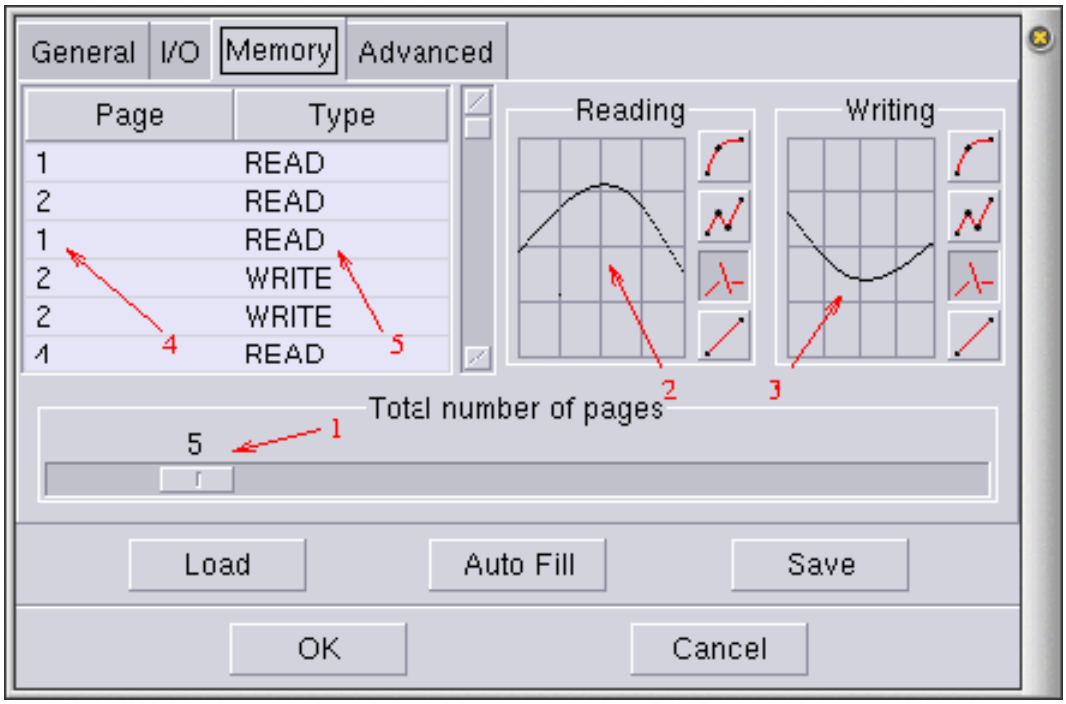



1. Add a new access.
2. Replace the currently selected access in the list.
3. Remove the currently selected access from the list.
4. Block which the access refers to.
5. Time at which the access will happen. This time refers to time spent by the process, not to system time.
6. Block which will be used when adding or replacing an I/O access.
7. Time which will be used when adding or replacing an I/O access.
8. The block for an access can also be selected clicking here.

## Memory

This are the accesses to main memory of the process, through out all it’s lifetime.

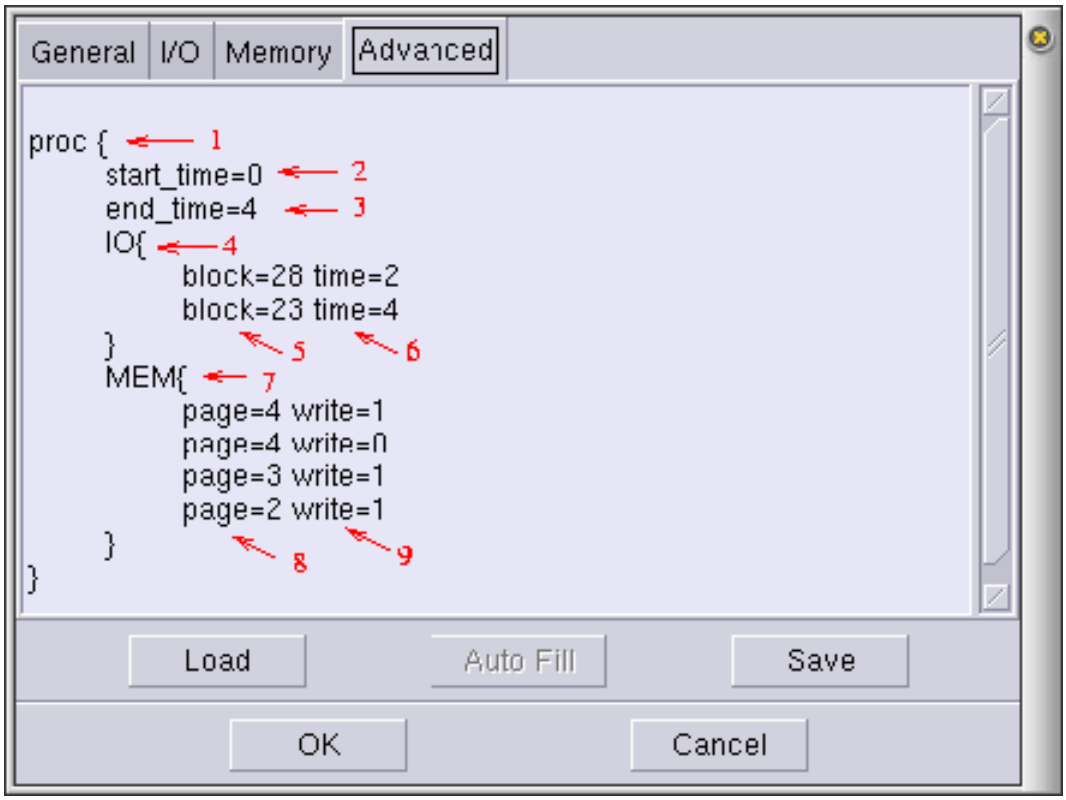
As it would be a pain to write all this by hand, the only way to fill it is choosing read and write probabilities on the graphs and clicking on "Auto Fill" button.




1. Total number of pages that this process will use.
2. Probability of the process reading each of its pages. X axis represents page numbers and Y axis represents probability.
3. Probability of the process writing to each of its pages. X axis represents page numbers and Y axis represents probability.
4. Page of the access.
5. Type of access, it may be READ or WRITE.

## Advanced

It is also possible to manually define all properties of the process with a simple language, the same used to write processes to file.

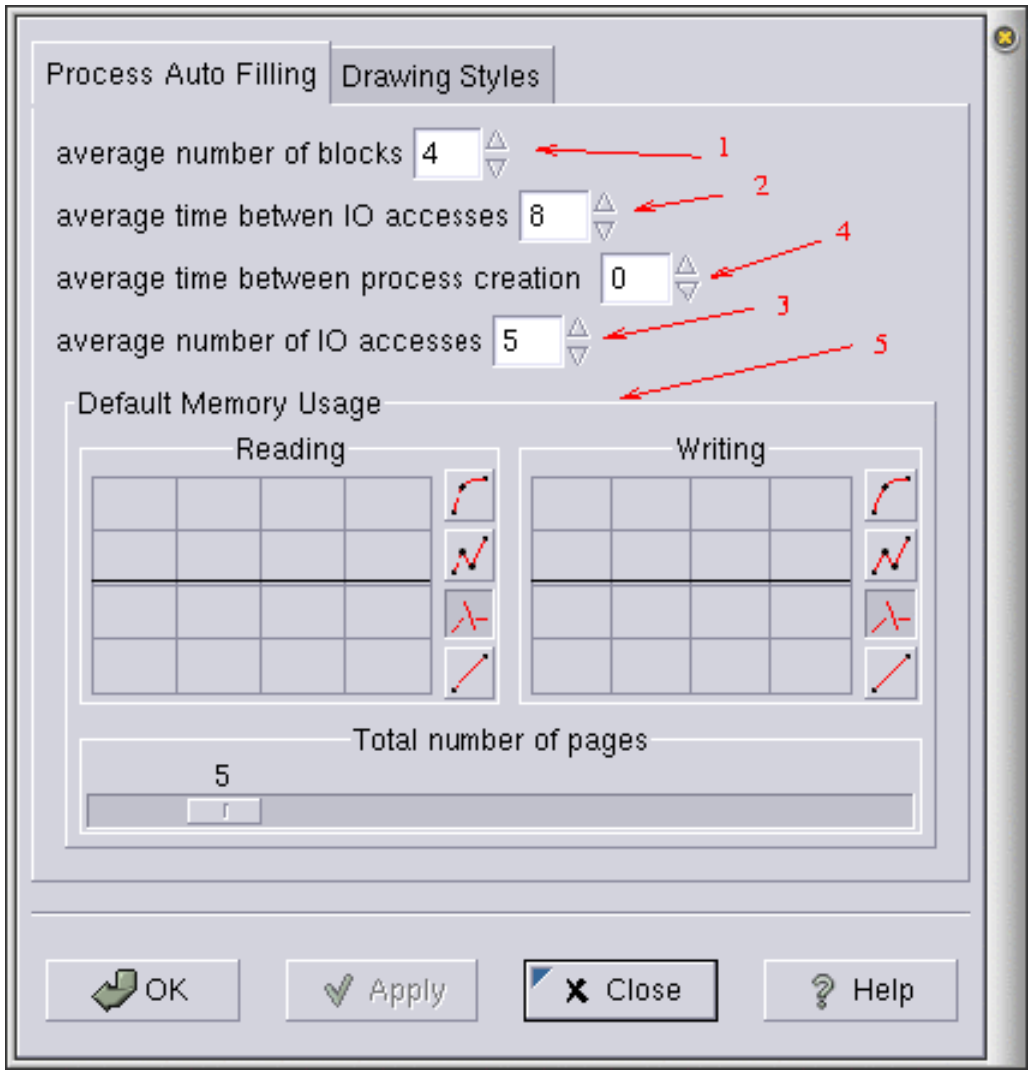



1. All information for the process is enclosed in the block "proc".
2. Time at which the process is inserted into the system.
3. Total process lifetime.
4. The list of disk accesses is enclosed within the block "IO".
5. Block used in this access.
6. Time at which the block is accessed, measured in process execution time, not global system time.
7. The list of memory accesses is enclosed within the block "MEM" and shown have an item for each "time unit" that the process runs.
8. Page for the access.
9. States weather the access is a read, "write=0", or a write, "write=1".

# Preferences

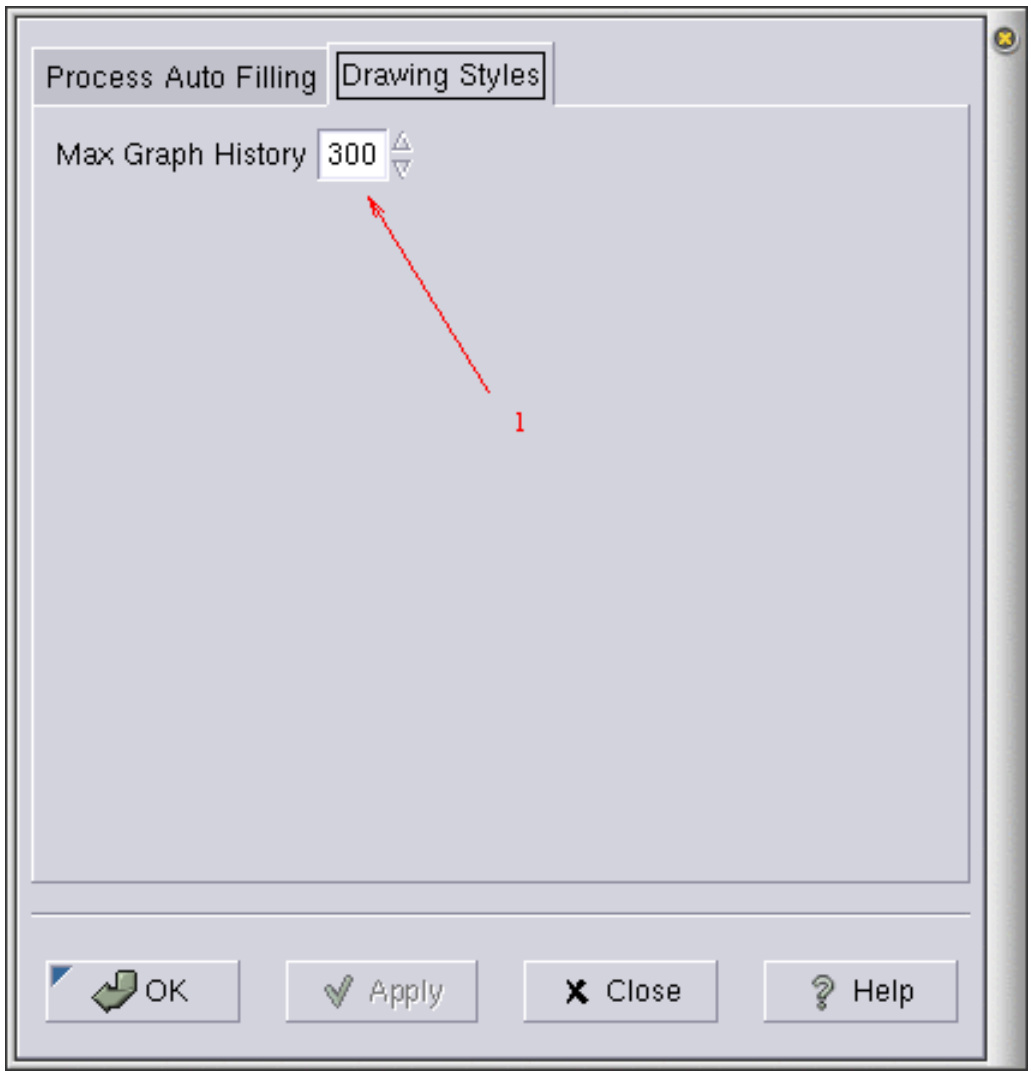
## Process Auto-Filling

This parameters are used to generate processes automatically.




1. Average number of blocks to be used by the process in total.
2. Average time between disk accesses.
3. Average number of memory accesses. If "1" is low and this parameter high, then there will be many accesses spread over a few blocks.
4. Average time between the creation of two consecutive processes.
5. Default values for memory access auto-filling.

### Drawing styles

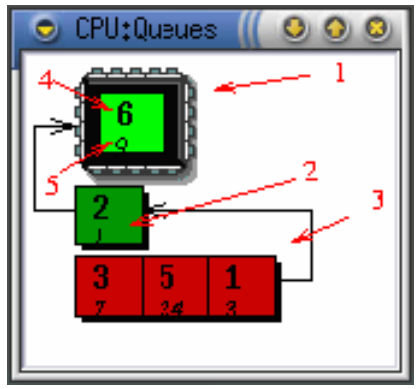



1. Limit for the drawing styles which maintain history of the system. This prevents VisualOS from eating up all resources over time.

## Drawing Styles

We will now see the different representations available for the processor and their meanings:

## Queues



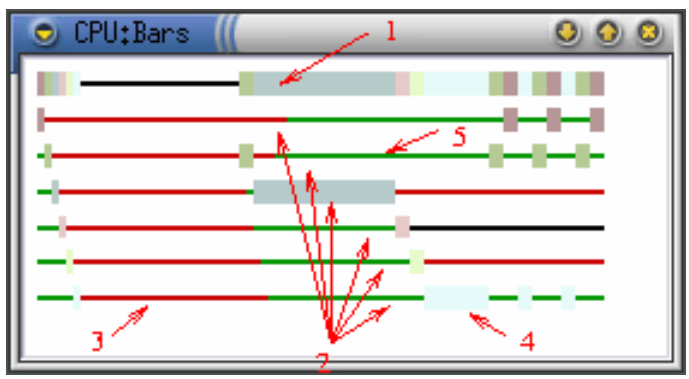

1. Currently running process.
2. "Ready to run" process queue; Multiple queues of this type are allowed depending on the algorithm in use.
3. Blocked process queue, waiting for a disk read or a page fault to complete.
4. Process identifier.
5. Remaining time until the next event which will block the process.

Processes may be selected clicking on them with the mouse.

## Bars

A different color is assigned to each process, and each one is represented in a different horizontal bar. The upper bar is a resume of all processes.

When a process is running, a thick line is drawn with it’s color on it’s bar and the upper bar. And when it is not running, a thin line is drawn, a red one if the process is blocked, a green one if it is "ready to run" and a black one if the process is finished.



1. Resume bar of all processes.
2. Bars of each process.
3. Blocked process.
4. Running process.
5. "Ready to run" process.

Processes may be selected clicking on their bar.

## Overlapped Bars

A different color is assigned to each process, and each one is represented in a different horizontal bar.

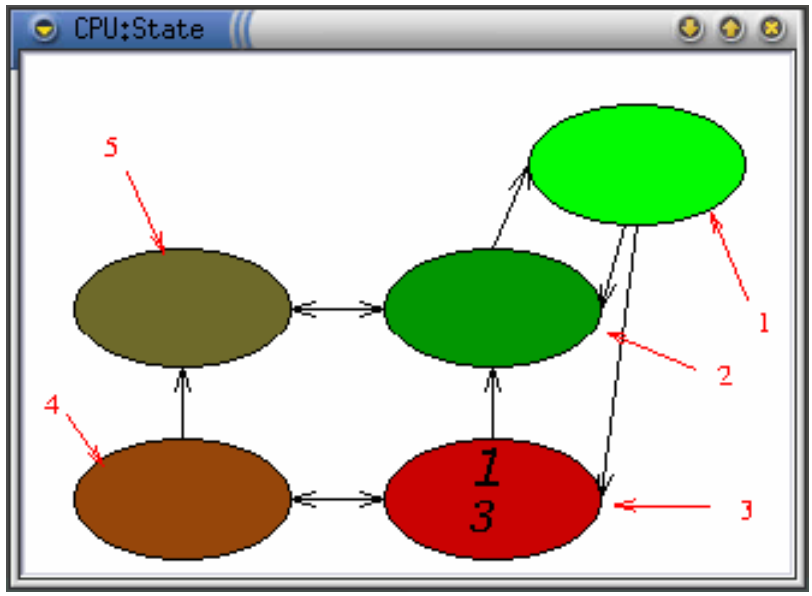
When a process becomes ready a thick line with it’s color is drawn on its bar, representing the CPU burst which the process pretends to run.


1. Burst which process 3 pretends to run.

Processes may be selected clicking on their bar with the mouse.

## Process status

Shows the state of the currently selected process.



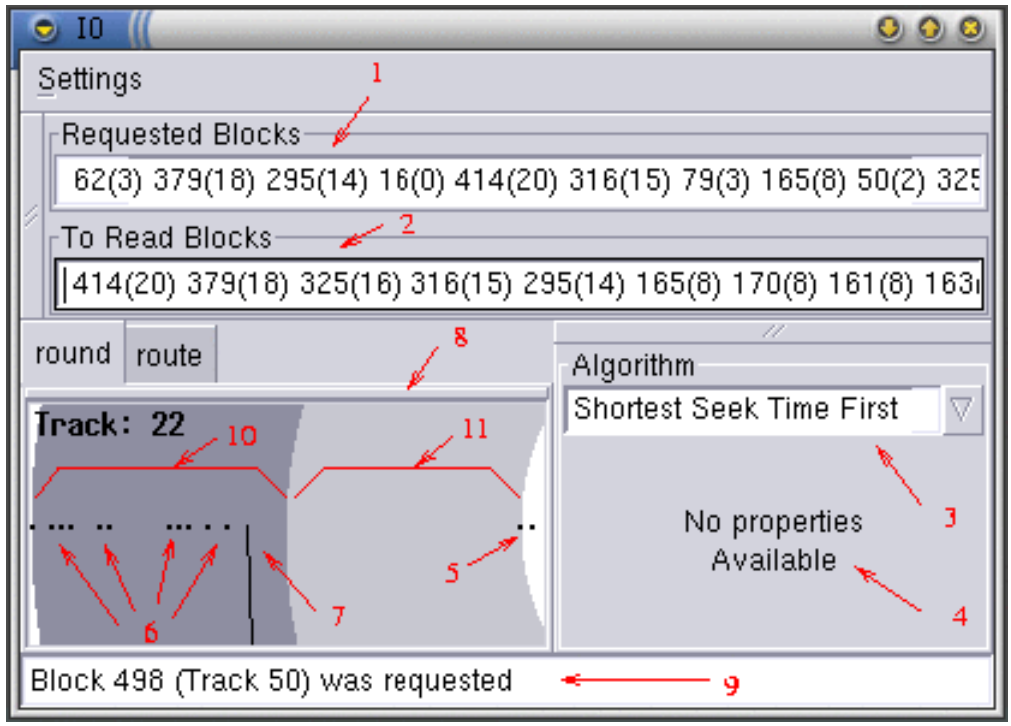

1. Running process.
2. "Ready to run" process.
3. Process blocked waiting for some event.
4. Blocked and suspended process.
5. "Ready to run" and suspended process.

In the picture, process 1 is blocked.

# Chapter 3. Secondary memory.

The input/output subsystem has the only function of: accept access requests, simulate the physical access and informing it's client (usually the CPU) when an access is completed.

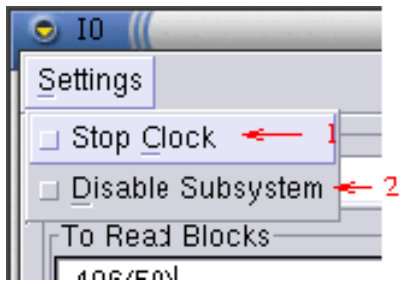
We will now describe it's elements:




1. List of requested blocks in order of arrival.
2. List of requested blocks sorted by the current algorithm.
3. Currently selected algorithm.
4. Currently selected algorithm's parameters. None, in this case.
5. Blocks requested from the swap area.
6. Blocks requested from the data area.
7. Reader head location.
8. Click on this button to put the subsystem representation in it's own window, which allows multiple simultaneous representations of the same subsystem to be visible at the same time.
9. Messages telling interesting events.
10. Data area.
11. Swap area.

## The menus

Here we will see what the options on each menu mean:

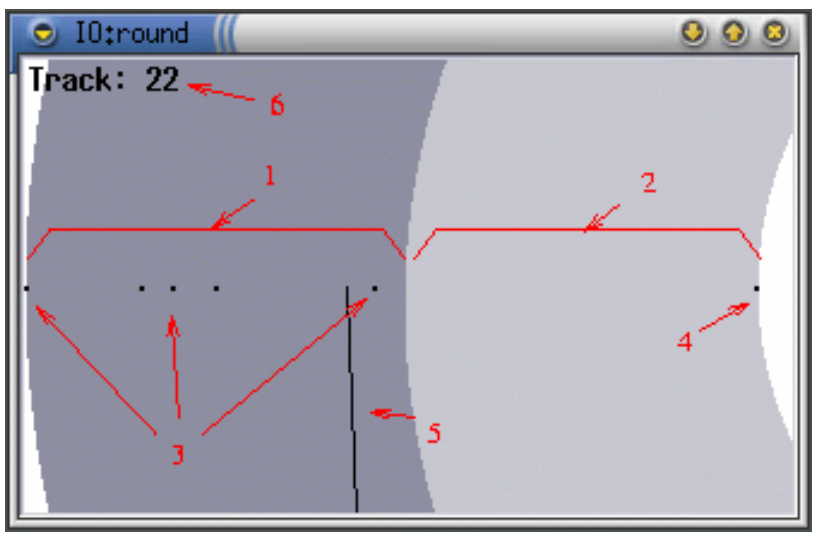



1. When this is checked, the clock will be stopped whenever something interesting happens.
2. When this is checked, I/O accesses will complete immediately, allowing the user to concentrate in some other aspect of the system.

## Drawing styles

We will now see the different representations available for the I/O subsystem.

### Round

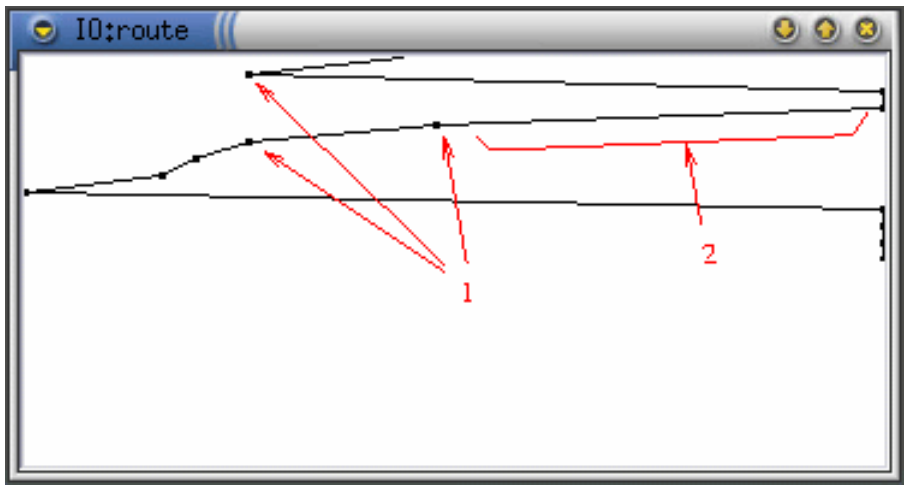



1. Data area.
2. Swap area.

3. Blocks requested from the data area.
4. Blocks requested from the swap area.
5. Reader head.
6. Current placement of the reader head.

## Route

This style, pretends to clearly show the trajectory which will follow the reader head to access all requested blocks depending on the selected algorithm.

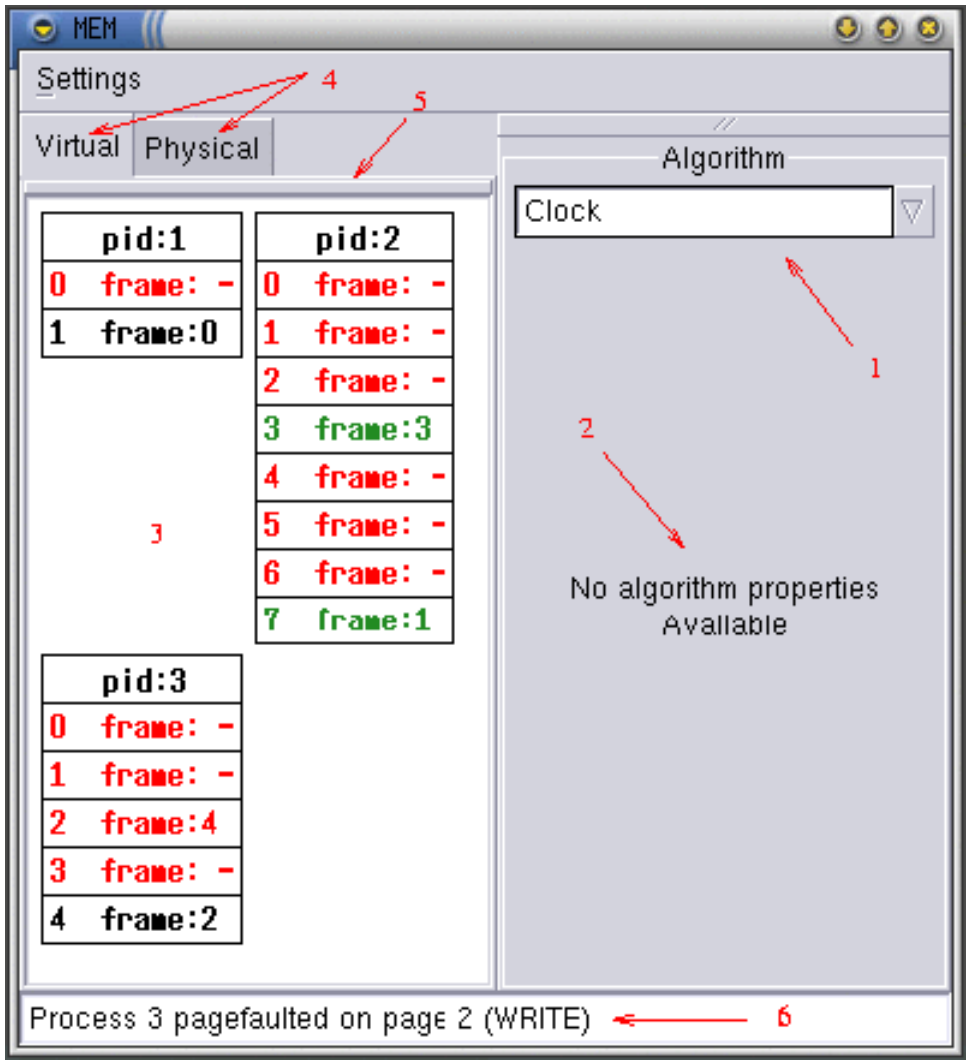



1. Requested Blocks.
2. Trajectory between block reads.

# Chapter 4. Main memory.

The memory subsystem has the only function of accepting memory usage requests and assigning physical memory frames to processes.

We will now describe it’s elements:




1. Currently used algorithm.
2. Parameters for the currently used algorithm. In this case none.
3. One of the possible representations.
4. Available representations on this subsystem.

5. Click on this button to place the representation on it's own window. This way you can see more than one simultaneously.
6. Messages telling interesting events.

## The menus

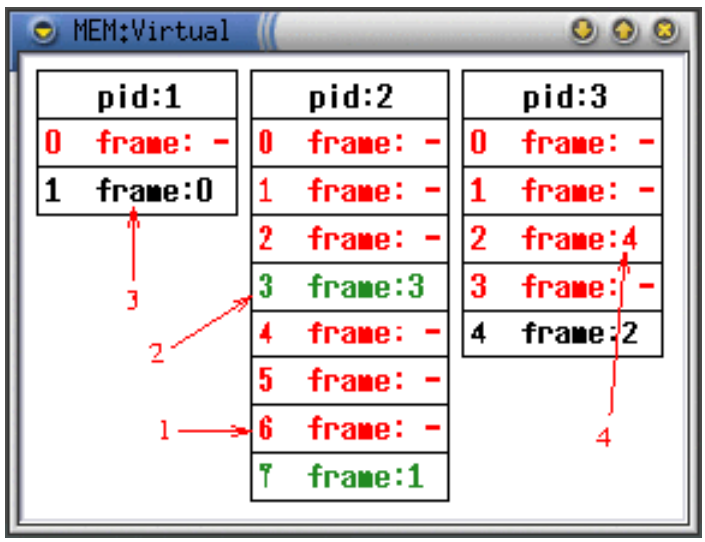
Here we will see what the options on each menu mean:


1. If enabled, the clock will be stopped whenever something interesting happens on this subsystem.
2. If enabled, this subsystem will pretend that all processes have all their pages; Something like having infinite memory. This is useful to concentrate on other aspects of the system.

## Drawing styles

We will now see the different representations available for the Memory subsystem.

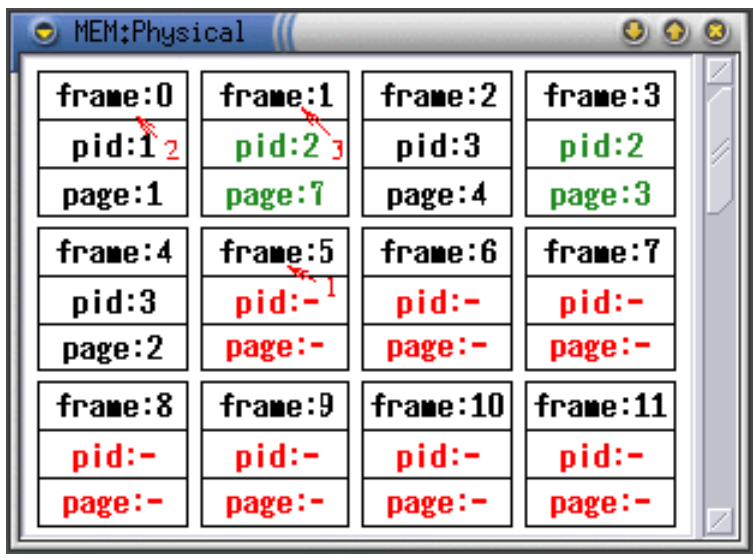
### Virtual Memory



1. Not valid page. Any access will cause a page fault.
2. Valid page. This page has not been modified since it was loaded into memory or since the last time it was written to swap space. So it could be discarded if needed and it's frame given to another page.
3. Valid page. This page has been modified since it was loaded into memory or since the last time it was written to swap space. So it we need the memory frame for a different page we would have to write it to swap space before reassigning it.
4. This page is not valid, but a frame as been assigned to it and will soon be valid.

## Physical Memory

This represents physical memory frames which get assigned to process' pages.

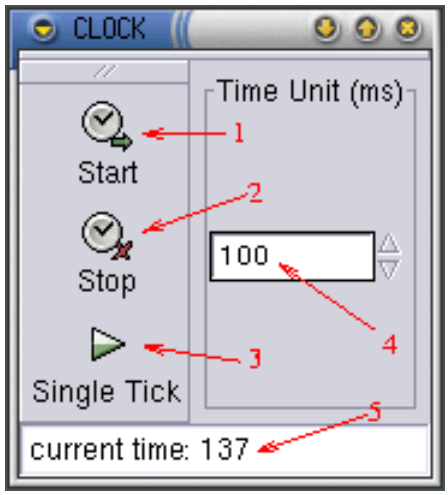



1. This frame is free.
2. This frame is assigned to process 1 and has been modified since it was loaded, so it will have to be written in swap space before it can be reused for a different page.
3. This frame is assigned to process 2 and was not modified since it was loaded, so it may be discarded without writing it to swap space if needed.

# Chapter 5. The Clock

The clock subsystem has the only task of controlling time, making other subsystems work faster, slower or stopping them.

We will now describe it’s elements:



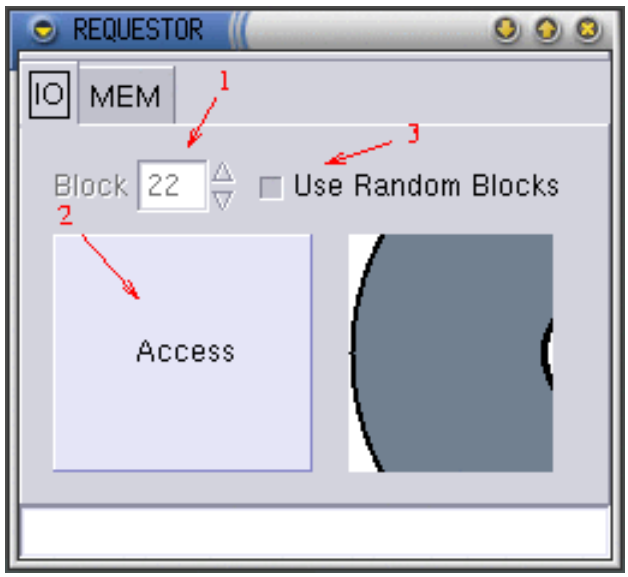

1. Clicking on this button the clock will start counting one time unit every time the selected milliseconds go by.
2. Clicking on this button the clock will stop.
3. Clicking on this button the clock will count a single "time unit".
4. This is the number of milliseconds equivalent to a "time unit", a lower number makes the time go faster and a higher number makes time go slower.
5. The number of "time units" elapsed until now.

# Chapter 6. The Requestor

This subsystem has the only task of making requests to main memory and I/O subsystems. This way you don’t have to create processes to play which those subsystems when we don’t care about the CPU.

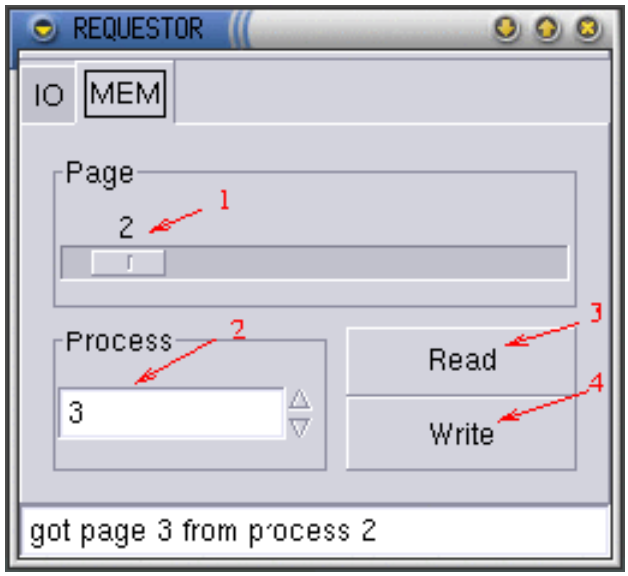
We will now describe it’s elements:

## Requests to main memory




1. This is the block which will be requested.
2. Clicking here the chosen block will be requested.
3. This enabled, random blocks will be requested when clicking the button.

## Main memory requests




1. We will use this page.
2. The request will apply to this process' virtual memory space.
3. Clicking here a read request is made.
4. Clicking here a write request is made.