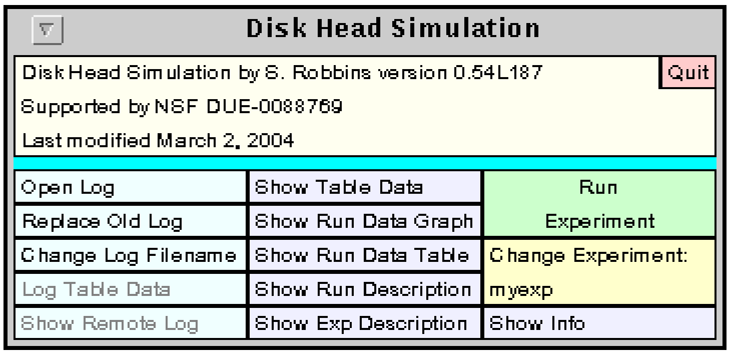
Experto A

Basic Operation

Execute `rundisk`. After a few seconds, you should see a small window appear similar to the one below.



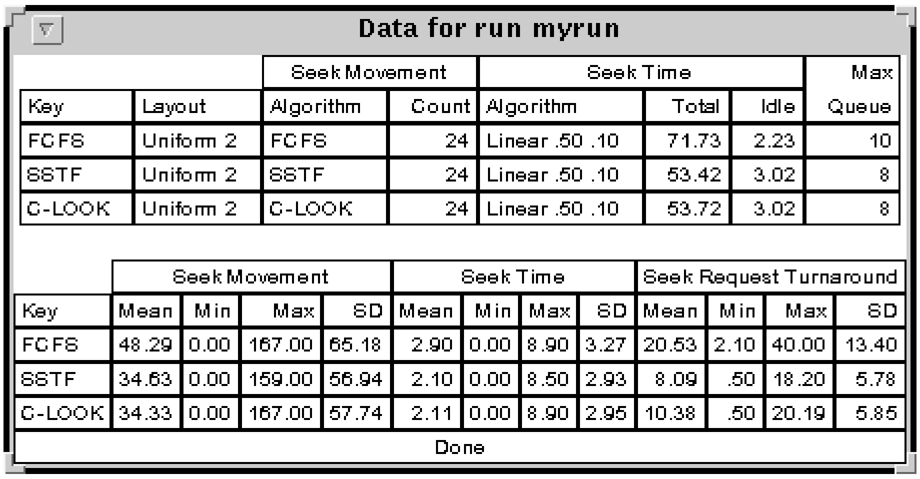
Disk Head Simulation by S. Robbins version 0.54L187
Supported by NSF DUE-0088769
Last modified March 2, 2004

Figure 1: The main simulator window.

The main simulator window has four parts.

1. The first three lines in the window give version information.
2. The **Quit** button in the upper right corner exits the simulator.
3. The thin bar under the version information is a progress bar that shows the progress of the simulator. When the experiment is run, the upper progress bar (in red) gives the progress of the experiment and the lower bar (in blue) gives the progress of the current run.
4. The bottom of the main window is taken up with buttons that control the simulator.
    - o The leftmost column of buttons control the logging features.
    - o The middle column displays information about the current experiment and results of the experiment.
    - o The last column is used to run the experiment.

When you push the large green **Run Experiment** button it splits into two buttons: **Pause Experiment** and **Abort Experiment** until the experiment is complete. When the experiment has completed, the button changes back to its original form. After running the experiment, the results can be displayed using the **Show Table Data** at the top of the middle column. This displays information about the result of the runs. Figure 2 shows the table for three runs.



		Seek Movement		Seek Time			Max
Key	Layout	Algorithm	Count	Algorithm	Total	Idle	Queue
FCFS	Uniform 2	FCFS	24	Linear .50 .10	71.73	2.23	10
SSTF	Uniform 2	SSTF	24	Linear .50 .10	53.42	3.02	8
C-LOOK	Uniform 2	C-LOOK	24	Linear .50 .10	53.72	3.02	8

	Seek Movement				Seek Time				Seek Request Turnaround			
Key	Mean	Min	Max	SD	Mean	Min	Max	SD	Mean	Min	Max	SD
FCFS	48.29	0.00	167.00	65.18	2.90	0.00	8.90	3.27	20.53	2.10	40.00	13.40
SSTF	34.63	0.00	159.00	56.94	2.10	0.00	8.50	2.93	8.09	.50	18.20	5.78
C-LOOK	34.33	0.00	167.00	57.74	2.11	0.00	8.90	2.95	10.38	.50	20.19	5.85

Figure 2: The statistics generated by the simulator.

Each run is labeled by its key. The top part of the table shows the three algorithms used by this run, the layout algorithm, the seek movement (or head) algorithm and the seek time (or seek movement) algorithm. Also shown are the number of seeks (Count column), the total and time seeking, the total idle time (when no head movement is occurring) and the maximum queue length. The bottom part of the table gives detailed numeric data about the seek movement (number of tracks moved per seek), the seek time, and the seek request turnaround time. This latter time is the time between when the seek request is make and when the seek is completed. For each of these four statistics, the main, minimum, maximum and sample standard deviation are given. The **Done** button at the bottom of the window is used to close the window.

The **Show Run Data Graph** displays a graphical representation of one of the runs. If more than one run has been made, you will be presented with a list of runs to choose from. After choosing one of these, a graph of the selected run will appear. Figure 3 below shows a graph for the SSTF algorithm.

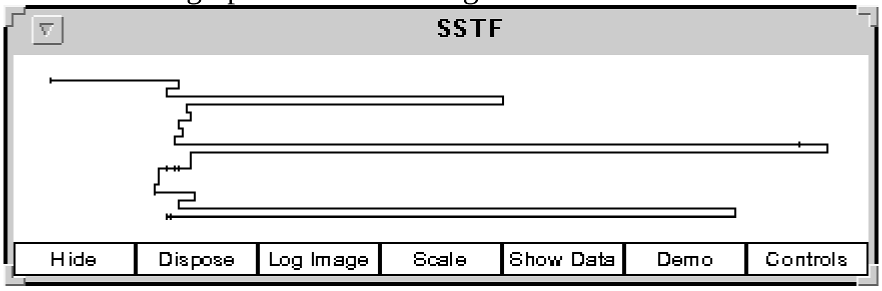


Figure 3: A graphical representation of the SSTF algorithm.

The horizontal direction corresponds to head position. Each time the head changes direction, the trace moves down. A tic mark appears each time a seek position is reached that does not cause a change in direction. The buttons on the bottom of the provide some additional features. The **Hide** button temporarily hides the window. It can be brought back again using the **Show Run Data Graph** button. The **Dispose** button permanently destroys the window and all its resources. If the **Show Run Data Graph** button is used again for this run, a new window is created. If logging is active, the **Log** button will put a copy of this image (without the buttons) in the log file. The **Scale** button will attempt to rescale the graphics so that it entirely fits in the window. The **Show Data** button pops up the same table available through the **Show Run Data Table** of the main simulator window. The **Demo** button enters **Demo Mode** for this window. Lastly, the **Controls** window allows you to set the scale and colors of various features of this window.

The **Show Run Data Table** button shows a detailed table of data about all of the seeks made in the given run. For each seek it shows the starting and ending block, the time the request was made, the time the seek started, the time the seek ended, the time to do the seek (the difference of the previous two entries), the turnaround time (between request and completion) and the size of the pending queue when the request was made. At the bottom of the table, the total number of seeks for this run is shown. Two button at the bottom of the table allow you to hide or dispose of the window.

The **Show Run Description** button gives a description of the runs from the current experiments. This is essentially the contents of the run file modified by the modifications in the exp file. The **Show Exp Description** button shows the contents of the exp file for the current experiment. The **Show Info** button displays a window giving a log of the runs made. It logs events (with wall clock time) such as the start and completion of each run.

Experto B

Configuration

The default configuration file is `diskheadconfig` but the simulator will use another file if it is specified on the command line when the simulator is started. A configuration file contains lines of keyword-value pairs. Lines beginning with a `%` are treated as comments and are ignored. The following table lists some valid keywords and the corresponding values.

Keyword	Values	Meaning
user	anything	The name of the current user. This appears in the log file.
quiet	none	Turns off all sounds generated by the simulator.
run	a filename	This is the name of a file with extension .run. The extension is not listed. Each run file used by the simulator must have a corresponding run line in the configuration file.
exp	a filename	This is the name of a file with extension .exp. The extension is not listed. Each exp file used by the simulator must have a corresponding run line in the configuration file.

Runs and Experiments

A run (or more precisely and experimental run) consists of a running the simulator with a specific set of algorithms and input data. Three algorithms are needed to specify a run.

- The movement algorithm describes how long it takes to move from one track to another. Currently, the only supported movement algorithm is **linear**. A linear movement algorithm is specified by a constant, `c`, and a rate, `r`. The time to move the head a total of `n` cylinders is `c + nr`.
- A layout algorithm which indicates how sectors are laid out on the disk. The simplest layout is **uniform** in which in cylinder has the same number of sectors.
- A head algorithm specifies how to determine which block to seek to next. Supported algorithms include FCFS, SCAN, C-SCAN, LOOK, C-LOOK, and SSTF.

Experimental runs are described in a file with extension `.run`. This file consists of lines, each containing a keyword followed by a value or values. The first line of the file should contain the keyword **name** followed by the name of the file without the `.run` extension. The second line should contain the keyword **comment** followed by a comment describing the run. This line is not used by the simulator. The following lines of the run file contain the following information:

Keyword	Values
key	Text describing the run. This is used in many of the tables to distinguish this run from others.
start	An integer representing a block number. This is the starting block number of the disk head. The default is 0.
layout	The value has one of two forms: uniform n, where n is an integer zoned n_1 m_1 n_2 m_2 ...

	The first form is for a uniform layout with n blocks per cylinder. The second form is for a zoned layout in which n_i is the number of blocks per sector and m_i is the number of sectors in the ith zone. This represents the physical layout of the disk, used to determining how long it takes to perform a seek. Each run file must have exactly one layout entry.
movement	This specifies how long it takes the disk head to move from one cylinder to another. The only currently supported movement algorithm is linear and specified by linear n m where n and m are integers. The time for a seek of x cylinders for x greater than 0 is n + mx. A seek of zero always takes 0 time. Each run file must have exactly one movement entry.
badfraction	The value is a floating point number between 0 and 1. It represents the fraction of references that are to bad blocks. A reference to a bad block is mapped to a location given by the baddestination keyword. It value defaults to 0.
baddestination	The value parameter is a distribution. This entry is necessary when badfraction is not 0. When a bad block is referenced, this distribution is used to generate the actual destination.
head	The value is one of the following specifications that determine when block to seek next: FCFS, SSTF, LOOK, CLOOK, SCAN n, CSCAN n
numblocks	The value is an integer specifying the number of block requests to generate from a distribution. The four keywords numblocks, firstarrival, interarrival and nextblock form a set and must all be present to specify a set of blocks to generate. Multiple sets many be specified.
firstarrival	The value is a floating point number representing the arrival time of the first block generated in this set.
interarrival	The value is a distribution specification giving the interarrival time of the requested blocks.
nextblock	The value is a distribution specification that is used for generating the next block request.

An experiment performs several runs and compares the results. An experiment is described by a file with extension .exp. Like the run file, it consists of lines containing keywords and values. The first two lines of an experiment file contain a name and comment field similar to those of a run file. The name line contains the name of the file without the `.exp` extension. The rest of the lines in an experiment file contain the keyword **run** followed by the name of a run file. The name of the run file may be followed by any number of keyword-value pairs that modify the run. The keywords and values are the same as those listed in the table. The run modifications allow the same run file to be used several times in the same experiment.

For example, suppose that the file `myrun.run` uses the FCFS head movement algorithm. The following file called `myexp.exp` will make three runs, all with the same parameters but using three different head movement algorithms:

```
name myexp
comment Three head algorithms compared
run myrun key "FCFS"
run myrun key "SSTF" head SSTF
run myrun key "C-LOOK" head CLOOK
```

The first run line has one modification, **key** while the others modify both the **key** and the **head**. Any modifications in the run line of an experiment file override the corresponding value in the named run file.

Experto C

Probability Distributions

Three probability distributions are supported at this time. When a probability distribution is specified in a file, it is represented by a single line of ASCII characters. The line starts with a word indicating the type of distribution and is followed by either a single floating point number representing the mean of the distribution, or in the case of the uniform distribution two floating point numbers representing the left and right endpoints of the interval. The following distributions are supported:

- The constant distribution. Example:
  constant 23.45
  represents a constant distribution with constant value 23.45.
- The exponential distribution. Example:
  exponential 23.45
  represents the exponential distribution with mean 23.45.
- The uniform distribution. Example:
  uniform 23.45 47.89
  represents the uniform distribution in the interval [23.45,47.89]

Head Algorithms

The term *head algorithm* refers to the method the simulator uses to decide which block to seek next. At any given time there are a number of pending block requests is a buffer or list. Each algorithm is specified by a short keyword. Some of these require one or more arguments. The following head algorithms are supported by the simulator:

Algorithm	Argument types	Description
SCAN	integer	The SCAN algorithm starts moving the head toward higher cylinder numbers until it reaches the last cylinder (given by the argument) stopping at pending blocks along the way. It then reverses direction and stops at pending blocks on the way back, moving al the way to cylinder 0.
CSCAN	integer	This is similar to SCAN but does not stop at pending blocks on the way bak to cylinder 0.
LOOK	none	LOOK is like SCAN, but it changes direction when there are no more pending blocks in the current direction.
CLOOK	none	This is like CSCAN but it changes directions when there are no more pending references.
FCFS	none	This seeks blocks in the order in which they are requested.
SSTF	none	The next block to seek is on the cylinder closest to the current one.

Movement Algorithms

The movement algorithm determines how long it takes to move the head from one track to another. The only supported movement algorithm is **linear** in which the time to move `n` tracks is `a + bn`. This formula is used whenever `n > 0`. The seek time for a seek of length 0 is always 0.

Layout Algorithms

The layout algorithm describes how sectors are laid out on the disk. There are two supported layouts: uniform and zoned. A uniform layout has a fixed number of blocks per cylinder. Since the simulator does not take into account rotational latency, the number of surfaces is not relevant. A uniform layout is specified as: `layout uniform n` where `n` is an integer greater than zero representing the number of sectors (blocks) per cylinder. In a zoned layout, the disk is divided into zones, each one having a uniform layout. A zoned layout is specified by giving the number of zones, `n` followed by `n` pairs of integers, each giving the number of cylinders in that zone and the number of blocks per cylinder for that zone. For example, `layout zoned 3 10 8 10 16 10 24` would represent a disk with 3 zones, of 10 cylinders each. The first 10 cylinders have 8 blocks per cylinder, the next 10 have 16 blocks per cylinder and the last 10 have 24 blocks per cylinder.

Specifying Blocks to Seek

The head algorithm is used to decide which block to seek next from a list of pending block requests. The simulator uses two methods for generating these block requests. For a short list of requests, the simplest method is to just specify each requested block number along with the time at which the request is made. An alternative to presenting a list of block numbers is to have the simulator generate block numbers using distributions. A collection of block numbers is specified with distributions by specifying four keyword-value pairs:

Keyword	Value
numblocks	An integer specifying how many block requests will be generated.
firstarrival	A floating point value indicating the time at which the first of these blocks arrives.
interarrival	A distribution giving the interarrival times of the generated blocks.
nextblock	A distribution used to generate the block request.

The four keyword-value pairs may occur in any order. Multiple distributions of block requests can be used by repeating these four keyword-value pairs.

Specifying Bad Blocks

Some disk drives automatically map bad sectors into sectors reserved for that purpose. When this happens, the operating system thinks it is accessing a block in a location different than the actual location on the disk. This has two consequences. The seek may take a different amount of time than expected by the operating system and the choice of the next block to seek will be based on incorrect information. The simulator can simulate this with two parameters, the **badfraction** and the **baddistribution**. The badfraction is the faction of block references that are to bad or mapped blocks. The baddistribution tells where these bad block are mapped.

Experto D

Logging

The simulator can keep a log file in HTML format that can be displayed and printed from a standard browser. The log file is controlled by the column of 5 buttons on the left side of the simulator main window.

The name and location of the log file is controlled by **logfile** and **logdir** configuration parameters. The simulator will store images in the logging directory with a name based on the **imagename** configuration parameter.

You can change the name of the log and image files after the simulator starts by using the **Change Log Filename** button. This will allow you to change the logging directory, the filename and the image filename.

Normally, when you open a new log file it will replace an existing log file of the same name. Pushing the **Replace Old Log** button changes it to **Append To Old Log** and causes future opens to append to and existing log file of the same name. It also modifies the image file name so that images created do not interfere with those created previously.

Open the log file by pushing the **Open Log** button. This changes the **Open Log** button to **Close Log** so that the log can be closed. It also changes the the buttons to **Log Comment** and **Stop Log**. The **Log Table Data** is activated.

The **Log Comment** button allows you to insert a comment into the log file. The **Stop Log** button changes to **Start Log** when pushed and allows you to temporarily stop and restart automatic logging without closing the log file.

The **Log Table Data** button inserts a table of data into the log file. The information in this table is similar to the information displayed with the **Show Table Data** button.

Demo Mode

The run data graph display for each run contains a **Demo Mode** button. When this button is pushed, demo mode for this display is entered. Any number of graphs for the same experiment can be simultaneously in demo mode. When this mode is active, a demo control window appears, similar to the one below.

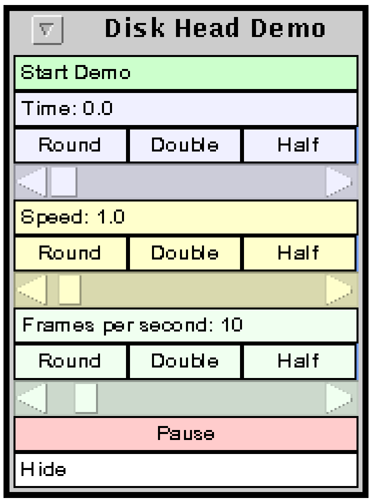


Figure 4: The control panel for demo mode.

This control panel allows you to control the simulator time. It contains a **Start Button**, a **Time Slider**, a **Speed Slider**, an **Update Rate** slider, and **Pause/Resume** button and a **Hide** button. Each of the sliders controls a particular quantity and consists of three rows of widgets of the same color. The first widget shows the value of the quantity being controlled. The second row contains three buttons. The first button rounds the value. The **Double** and **Half** buttons control the upper limit of the slider value. The third row contains the slider.

You can either set it manually by moving the time slider or to automatically run the time at a fixed rate by pushing the **Start Demo** button. The speed at which time progresses is controlled the **Speed Slider**. How often the graphs update is controlled by the last slider, labeled **Frames per second**. Next there is the **Pause** button. When it is pushed, the simulation time stops and the button changes to a **Resume** button. Lastly, the **Hide** button hides the demo control window. You can bring this window back by turning off and on the demo in any of the run data graph displays.

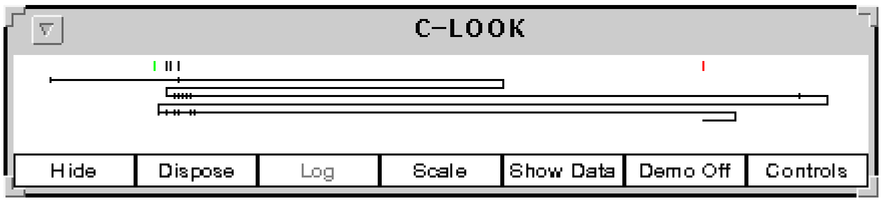


Figure 5: Demo mode.

Figure 5 shows graph displays of one algorithm in demo mode along with the demo control panel. The time is paused at 44.695. The graph shows the state of the simulation at that time. At the top of the graph are tick marks in black, green and red. The black marks are pending requests. The request of the current seek is shown in green. The current head position is shown in red.