Devices, Input, and Output
Example: Disk Device Driver

• Must provide an illusion of a linear array of sectors that are numbered like elements of an array

• Sector $s$ on track $t$ in cylinder $c$ is numbered

\[ a = ((c \times \text{(#tracks/cylinder)} + t) \times \text{(#sectors/track)}) + s \]

so rather than referring to $(c, t, s)$, kernel can refer to $a$

• Also must reduce effect of latencies of accessing disk
  • Overlap I/O and computation
  • Arrange large objects only one seek is needed to read/write them
  • Order outstanding disk requests
Ordering Disk Requests: Assumptions

• Only 1 disk drive
• All I/O requests are for single equal-size blocks
• Requested blocks distributed over disk
• Disk has only 1 moveable arm with all heads on it
• Seek latency is linear in the number of tracks crossed
  • Ignores disk controller using sectors from tracks at end of disk to replace bad sectors
• Disk controller does not introduce appreciable delays
• Read, write requests are equally fast
Key Points for Evaluating Disk Access Policies

• How long must requests wait as a function of load
  • Frequency of requests, measured in requests per time unit
• Mean, variance of waiting time for each request
  • Low mean, high variance means some requests will take a long time
Disk Access Policies

• First come, first serve (FCFS)
• Pickup
• Shortest seek time first (SSF, SSTF)
• SCAN
• N-Step SCAN
• C-SCAN
First Come First Serve (FCFS)

• Requests cannot be starved; all get serviced eventually
• Fairly low variance, but becomes saturated easily
  • Load becomes greater than driver can handle, so requests always waiting
• Problems
  • Every request is likely to require a seek
  • Great for low loads, but for high loads the latencies increase the mean of waiting time
Pickup

- FCFS but, as the head moves to the next track, any queued requests for the tracks it passes over are serviced
- For high loads, this decreases the mean waiting time a bit
Shortest Seek Time First (SSF, SSTF)

• Service the request lying on the closest track
  • Saturates at the highest load of any of these policies

• Problems:
  • Starvation; this means the disk can’t keep up with disk I/O requests, usually indicating a more severe problem such as thrashing
  • Variance larger than that of FCFS as innermost and outermost tracks are serviced less frequently than others
SCAN

• Head moves from outermost track to innermost, then back, etc., servicing requests along the way

• It reduces the problem of the innermost and outermost tracks getting less service
  • So it lowers the variance

• Problem: still subject to starvation
N-Step SCAN

• Like SCAN, but when a sweep begins (going in or out), the requests in the device queue *at that time* are the only ones serviced
• Arrivals after that wait for the next sweep to begin
• Starvation not possible
• Reduces variance even more
Circular SCAN (C-SCAN)

• Like SCAN, but requests are serviced only when the head is moving from outermost to innermost track
  • It “jumps back” from the innermost track to the outermost track
• Eliminates problem of innermost, outermost tracks getting less frequent service
• Waiting times also more uniform
LOOK variants

• With SCAN, heads always go to the innermost and outermost tracks, even if there are no requests for service involving those tracks
• LOOK variants have the heads go only as far as there are outstanding requests, and then have the head reverse direction
• Example: 200 track disk, requests for tracks 150, 90 and 70, with heads currently at 110 and moving inward
  • Handle request for track 90, then track 70
  • Change direction at track 70 rather than continue inward
Comparison

• Disk has 200 tracks (tracks numbered from 199 to 0)
• Head is currently at track 53
• Set of requests in the queue is
  98 183 37 122 14 124 65 67
• For LOOK and SCAN, assume head is moving inward
# Order of Service

<table>
<thead>
<tr>
<th>policy</th>
<th>order of servicing</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCFS</td>
<td>98 183 37 122 14 124 65 67</td>
</tr>
<tr>
<td>PICKUP</td>
<td>65 67 98 122 124 183 37 14</td>
</tr>
<tr>
<td>SSTF</td>
<td>65 67 37 14 98 122 124 183</td>
</tr>
<tr>
<td>SCAN</td>
<td>37 14 65 67 98 122 124 183</td>
</tr>
<tr>
<td>LOOK</td>
<td>37 14 65 67 98 122 124 183</td>
</tr>
<tr>
<td>C-SCAN</td>
<td>37 14 183 124 122 98 67 65</td>
</tr>
<tr>
<td>C-LOOK</td>
<td>37 14 183 124 122 98 67 65</td>
</tr>
</tbody>
</table>
Number of cylinders heads move over to service each request

<table>
<thead>
<tr>
<th>policy</th>
<th>number of cylinders moved over</th>
<th>total</th>
<th>mean</th>
<th>std dev</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>number</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FCFS</td>
<td>45  85  146  85  108  110  59  2</td>
<td>640</td>
<td>80.00</td>
<td>44.47</td>
</tr>
<tr>
<td>PICKUP</td>
<td>12  2  31  24  2  59  146  23</td>
<td>299</td>
<td>37.38</td>
<td>47.57</td>
</tr>
<tr>
<td>SSTF</td>
<td>12  2  30  23  84  24  2  59</td>
<td>236</td>
<td>29.50</td>
<td>28.62</td>
</tr>
<tr>
<td>SCAN</td>
<td>16  23  79  2  31  24  2  59</td>
<td>236</td>
<td>29.50</td>
<td>26.97</td>
</tr>
<tr>
<td>LOOK</td>
<td>16  23  51  2  31  24  2  59</td>
<td>208</td>
<td>26.00</td>
<td>20.72</td>
</tr>
<tr>
<td>C-SCAN</td>
<td>16  23  231  59  2  24  31  2</td>
<td>388</td>
<td>48.35</td>
<td>75.93</td>
</tr>
<tr>
<td>C-LOOK</td>
<td>16  23  169  59  2  24  31  2</td>
<td>326</td>
<td>40.75</td>
<td>54.89</td>
</tr>
</tbody>
</table>
## Waiting Time

Time each request has to wait for service, in terms of cylinders crossed

<table>
<thead>
<tr>
<th>policy</th>
<th>cumulative number of cylinders moved over</th>
<th>total</th>
<th>mean</th>
<th>std dev</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FCFS</td>
<td>45 130 276 361 469 579 638 640</td>
<td>3138</td>
<td>392.25</td>
<td>228.78</td>
</tr>
<tr>
<td>PICKUP</td>
<td>12 14 45 69 71 130 276 299</td>
<td>916</td>
<td>114.5</td>
<td>113.24</td>
</tr>
<tr>
<td>SSTF</td>
<td>12 14 44 67 151 175 177 236</td>
<td>876</td>
<td>109.50</td>
<td>85.60</td>
</tr>
<tr>
<td>SCAN</td>
<td>16 39 118 120 151 175 177 236</td>
<td>1032</td>
<td>129.00</td>
<td>73.12</td>
</tr>
<tr>
<td>LOOK</td>
<td>16 39 90 92 123 147 149 208</td>
<td>864</td>
<td>108.00</td>
<td>62.37</td>
</tr>
<tr>
<td>C-SCAN</td>
<td>16 39 270 329 331 355 386 388</td>
<td>2114</td>
<td>264.25</td>
<td>150.91</td>
</tr>
<tr>
<td>C-LOOK</td>
<td>16 39 208 267 269 293 324 326</td>
<td>1742</td>
<td>217.75</td>
<td>123.33</td>
</tr>
</tbody>
</table>
Optimizations

• Sector queueing
  • Policy to minimize rotational latency
  • Order requests for the same track so they can be serviced with a minimum number of rotations of disk
  • Implementation: each sector has its own queue for requests
  • Used only when there are extremely heavy loads

• Caching
  • Read extra sectors following the one you want
Process Interface

• Concept of file underlies interface
  • More about this next

• Enables processes to interact with devices
  • Also kernel structures such as /dev/null and /proc

• Need at least 1 special system call to handle device-specific functions
System Calls: *open*, *close*

- *open* makes file accessible to process
  - Form: descriptor = *open*(file, how, . . . )
    - Now process uses descriptor to refer the file
    - If device not ready, process may block or call may return error code
    - Call also checks privileges to ensure user can open the file
- *close* disassociates file from process
  - Form: *close*(descriptor)
    - Device driver does any needed clean-up
System Calls: seek

- *seek* positions pointer associated with descriptor as instructed
- **Form**: `seek(descriptor, where)`
  - Read/write pointer repositioned to where
  - Examples: go to arbitrary location in file, position on tape
- **Linux**: `lseek(descriptor, offset, whence)`
  - whence indicates if offset is from beginning or end of descriptor, or current position of read/write pointer
  - Returns new position on success, –1 on error; but –1 may be valid value
  - Disambiguate using *errno*
System Calls: seek

• Linux: *lseek* example

```c
external int errno;

... 
errno = 0;
if (lseek(desc, offset, SEEK_SET) == -1 && errno != 0){
    /* handle error */
}
else{
    /* handle success */
}
```
System Calls: \textit{read}

• Transfers data from descriptor object to memory

• Form: \texttt{nread} = \texttt{read}(descriptor, memory address, amount)
  • Reads \texttt{nread} bytes, which is at most \texttt{amount}
  • Returns 0 on end of file, error code on error

• Form: \texttt{nread} = \texttt{readv}(descriptor, memory list, list length)
  • Like \texttt{read}, but reads data into multiple memory locations
  • Locations given in memory list; also number of bytes for each
  • Returns number of bytes read, or 0 on end of file, error code on error
System Calls: **write**

- Transfers data from memory to descriptor object
- **Form:** \( nbyte = \text{write}(\text{descriptor}, \text{memory address}, \text{amount}) \)
  - Outputs \( nbyte \) bytes, which is at most \( amount \)
  - Returns error code on error
- **Form:** \( nbyte = \text{writev}(\text{descriptor}, \text{memory list}, \text{list length}) \)
  - Like write, but writes data from multiple memory locations
  - Locations given in memory list; also number of bytes for each
  - Returns number of bytes written, error code on error
Blocking vs. Non-Blocking Read and Write

• Blocking transfer is synchronous
  • So when the next statement is executed, transfer has been completed

• Non-blocking transfer is asynchronous
  • So next statement executed whether or not transfer has been completed

• Two ways to determine when non-blocking transfer completes:
  • Use polling by checking an indicator
  • Use interrupts
Non-Blocking Read and Write

• Process requests interrupt from kernel when transfer completes
  • System call may arrange this; on Linux, it’s SIGIO
• Process must arrange to catch interrupt and process it
  • Usually a system call like `handler(signal, function)`
• If process does need to block until transfer is complete, need a system call like `wait(descriptor, timeout)`
  • Blocks until transfer to or from descriptor completes
  • If not completed by timeout, then wake up and continue
• Never modify memory involved in transfer until transfer completes
  • Results are undefined
System Calls: control

• Used for device-specific actions
• Form: control(descriptor, action, . . . )
  • action is device specific and may require other parameters
• Linux example: make FAT file system read-only:
  attrmask = ATTR_RO;
  ioctl(desc, FAT_IOCTL_SET_ATTRIBUTES, &attrmask)
Linux Examples

• Insert ch into (terminal) input queue:
  \[ \text{toinsert} = \text{ch}; \]
  \[ \text{ioctl}(\text{desc}, \text{TIOCSTI}, \&\text{toinsert}) \]

• Give up role of controlling terminal:
  \[ \text{ioctl}(\text{desc}, \text{TIOCNOTTY}) \]