Discussion 8

File Systems

04/05/24

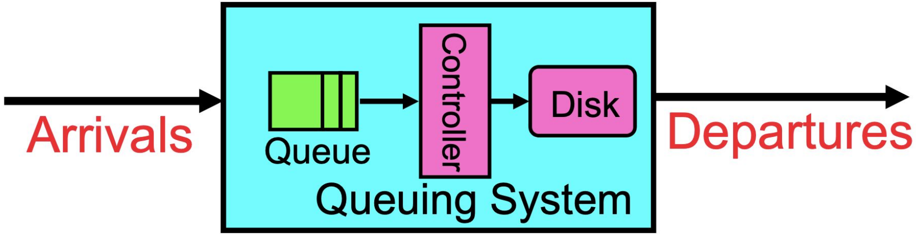
Staff

Announcements

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
			Homework 4 Due	Homework 5 Release	Project 2 Due	
Project 3 Release					RPC Lab Deadline	
	Project 3 Design Doc Due				HW5 Checkpoint Deadline	

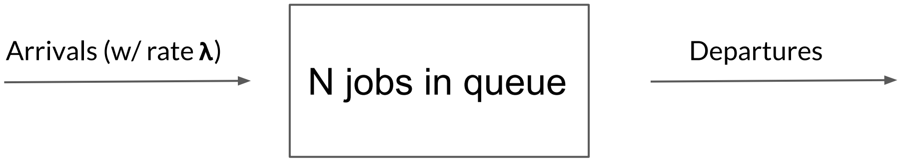
Queueing Theory

Introduction



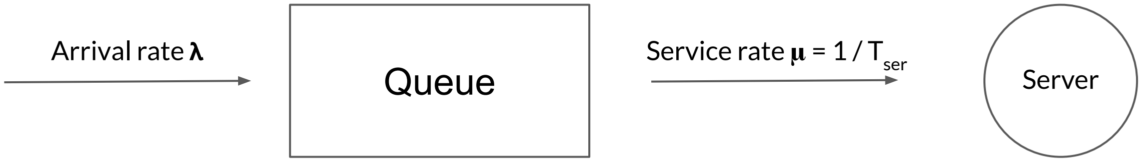
- Arrivals and departures are both characterized by some probabilistic distributions
- Queueing Theory applies to long-term, steady-state behavior
    - Arrival rate = Departure rate

Little’s Law



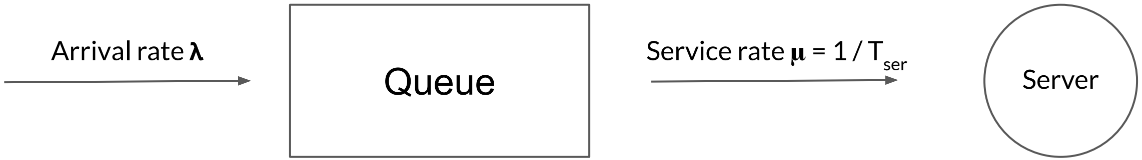
- In any **stable** system, average arrival rate = average departure rate
- Average arrival rate **λ**
- Average time spent in queue **L**
- Average number of jobs in queue **N** = **λ** x **L**
- This holds regardless of instantaneous variations (remember, this is long–term behavior)

Computing L (aka T_Q)



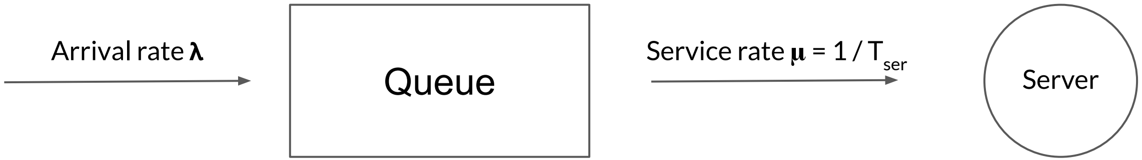
- To use Little's Law, we needed to know **L**, the average amount of time spent waiting in the queue
    - Notation: **L** is also often called $\mathbf{T_Q}$
- Parameters that define our system:
    - **λ**: mean arrival rate
    - T_{ser}: mean service time (time to service a job/customer/etc.)
    - $C = \boldsymbol{\sigma}^2 / T^2_{ser}$: squared coefficient of variation
        - Let **X** be the random variable representing service time
        - $C = Var(\mathbf{X}) / E[\mathbf{X}]^2$

Computing T_Q



- Parameters that define our system:
  - $\boldsymbol{\lambda}$: mean arrival rate
  - T_{ser}: mean service time (time to service a job/customer/etc.)
  - $C = \boldsymbol{\sigma^2} / T^2_{ser}$: squared coefficient of variation
    - Let $\mathbf{X}$ be the random variable representing service time
    - $C = Var(\mathbf{X}) / E[\mathbf{X}]^2$
- From these parameters, we can calculate the following useful intermediate values:
  - $\boldsymbol{\mu} = 1 / T_{ser}$: mean service rate
  - $u = \boldsymbol{\lambda} / \boldsymbol{\mu} = \boldsymbol{\lambda} \times T_{ser}$

Computing T_Q



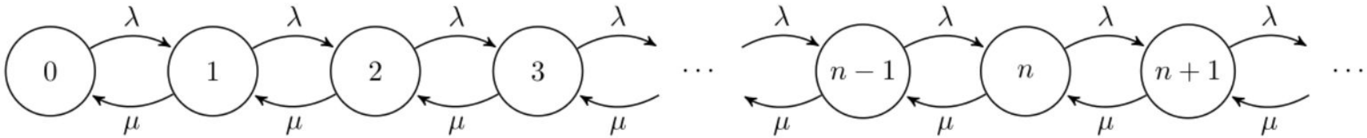
- Results:
  - Memoryless Service Time Distribution (M/M/1 Queue):
    - $C = 1$
    - $T_Q = T_{ser} \times u / (1 - u)$
  - General Service Time Distribution (M/G/1 Queue):
    - $T_Q = T_{ser} \times ½ (1 + C) \times u / (1 - u)$

Derivation of M/M/1 Queue formula (out of scope)

- Model the system as a CTMC with the transition matrix

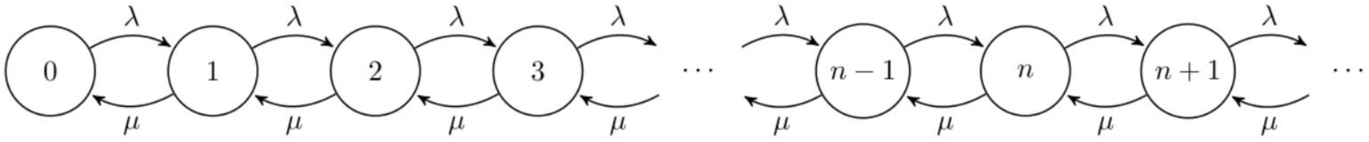
$$Q = \begin{pmatrix} -\lambda & \lambda & & & \\ \mu & -(\mu+\lambda) & \lambda & & \\ & \mu & -(\mu+\lambda) & \lambda & \\ & & \mu & -(\mu+\lambda) & \lambda \\ & & & & \ddots \end{pmatrix}$$

- This is just the following birth-death chain



- System is stable only when u = $\lambda / \mu < 1$.

Derivation of M/M/1 Queue formula (out of scope)



- The stationary distribution for number of jobs in system (queue + being served) is ~ Geo(1 - u), i.e. $P[X = k] = (1 - u) \times u^k$ where X is the number of jobs
- N, the average number of jobs in the system is simply $E[X] = u / (1 - u)$
- By Little's Law, $N = \boldsymbol{\lambda} W$
  - $\boldsymbol{\lambda} W = u / (1 - u)$ => $W = 1 / (\boldsymbol{\mu} - \boldsymbol{\lambda})$
- $T_Q = W - T_{ser} = [1 / (\boldsymbol{\mu} - \boldsymbol{\lambda})] - [1 / \boldsymbol{\mu}] = u / (\boldsymbol{\mu} - \boldsymbol{\lambda}) = (1 / \boldsymbol{\mu}) * \boldsymbol{\lambda} / (\boldsymbol{\mu} - \boldsymbol{\lambda}) = T_{ser} \times u / (1 - u)$

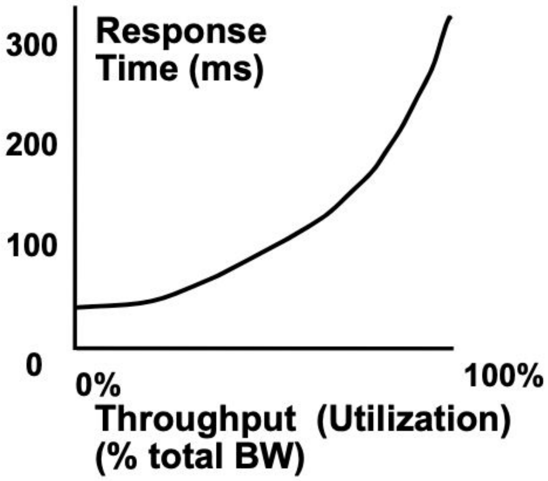
Concept Check

1. Explain intuitively why response time is nonlinear with utilization. Draw a plot of utilization (x axis) vs. response time (y-axis) and label the endpoints on the x-axis.

Concept Check

1. Explain intuitively why response time is nonlinear with utilization. Draw a plot of utilization (x axis) vs. response time (y-axis) and label the endpoints on the x-axis.

   As long as utilization is less than 100%, the server will be idle for some non-zero portion of the time. This idling time is wasted and can't be recovered ("anti-wasted"). Because of this asymmetry, the length of the queue builds up over time.



Concept Check

2. If 50 jobs arrive at a system every second and the average response time for any particular job is 100ms, how many jobs are in the system (either queued or being serviced) on average at a particular moment? Which law describes this relationship?

3. Is it better to have N queues, each of which is serviced at the rate of 1 job per second, or 1 queue that is serviced at the rate of N jobs per second? Give reasons to justify your answer.

Concept Check

2. If 50 jobs arrive at a system every second and the average response time for any particular job is 100ms, how many jobs are in the system (either queued or being serviced) on average at a particular moment? Which law describes this relationship?

   50 jobs/s × 0.1 s = 5 jobs (at any time). This is Little's law - arrival rate × average response time = average length of the queue.

3. Is it better to have N queues, each of which is serviced at the rate of 1 job per second, or 1 queue that is serviced at the rate of N jobs per second? Give reasons to justify your answer.

Concept Check

2. If 50 jobs arrive at a system every second and the average response time for any particular job is 100ms, how many jobs are in the system (either queued or being serviced) on average at a particular moment? Which law describes this relationship?

   50 jobs/s × 0.1 s = 5 jobs (at any time). This is Little's law - arrival rate × average response time = average length of the queue.

3. Is it better to have N queues, each of which is serviced at the rate of 1 job per second, or 1 queue that is serviced at the rate of N jobs per second? Give reasons to justify your answer.

   One queue that can process N jobs per second is faster (i.e the second option).
   Better response time (1 / N sec vs 1 sec) and better utilization (no load-balancing problems), which gives you lower queuing delays on average.
   With the single, fast queue, every arriving request gets the same fast average service. In the multiple-queue case, you only get the same service speed when all N servers are serving customers, i.e. only in the cases when a burst of items arrives quickly enough to get every server busy.

Concept Check

4. What is the average queuing time for a work queue with 1 server, average arrival rate of λ, average service time S, and squared coefficient of variation of service time C?

5. What does it mean if C = 0? What does it mean if C = 1?

Concept Check

4. What is the average queuing time for a work queue with 1 server, average arrival rate of λ, average service time S, and squared coefficient of variation of service time C?

   S is another name for T_{ser}.

   $T_Q = S \times \frac{1}{2}(1 + C) \times u / (1 - u)$ where $u = \boldsymbol{\lambda} \times S$

5. What does it mean if C = 0? What does it mean if C = 1?

Concept Check

4. What is the average queuing time for a work queue with 1 server, average arrival rate of λ, average service time S, and squared coefficient of variation of service time C?

   S is another name for T_{ser}.

   $T_Q = S \times ½ (1 + C) \times u / (1 - u)$ where $u = \boldsymbol{\lambda} \times S$

5. What does it mean if C = 0? What does it mean if C = 1?

   If C = 0, then your service rate is regular and deterministic, which means that tasks are completed at a constant rate. If C = 1, then your arrivals can be modeled as a Poisson Process, and the interval between jobs being serviced can be modeled as a exponential distribution.

File Systems

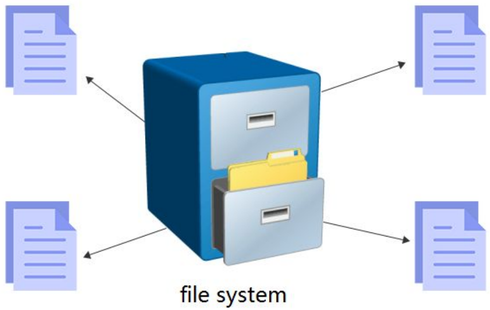
File System

File system provides persistent, named data to the user.

- Abstraction layer maintained by the OS to make it easy for user and programs to interact with files.

File is a named collection of data in a file system.

- Allow an arbitrary amount of data to be referred to by a single, meaningful name.
- **Metadata** contains information about a file needed by the OS (e.g. size, owner, access control).
- **Data** is the actual content of the file.
    - Can be interpreted to be something meaningful (e.g. text editors, PDF readers), but at its core is just a raw sequence of bytes.

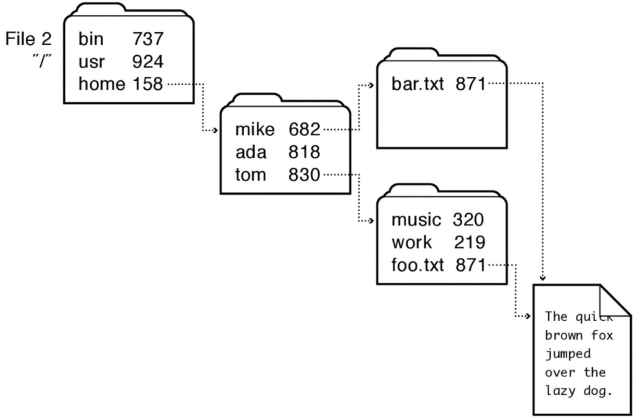


File System

A **directory** is a list of mappings from human-readable file names to specific underlying files or directories.

- **Hard links** are directory entries that map different names to the same file number.
  - Allow for one underlying file to have many names.
  - Can't span across multiple file systems.
  - Create using `ln [target] [dest]`.
- **Soft/symbolic links** are directory entries that map a name to another name.
  - Can move across different file systems.
  - Create using `ln -s [target] [dest]`.

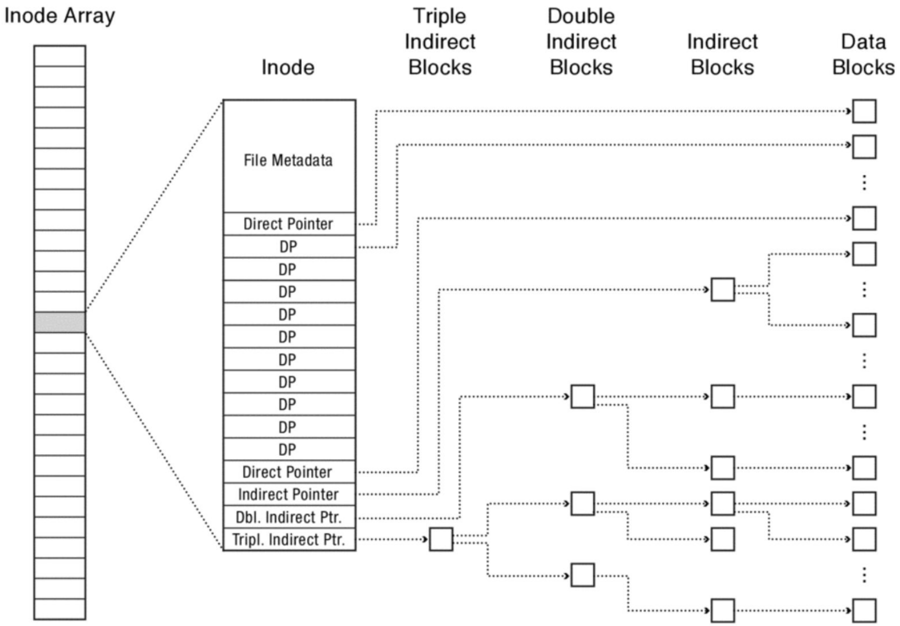
When analyzing file system implementations, focus on index structure, directory structure, and free space management.



Fast File System (FFS)

Index structure

- Fixed, asymmetric tree called a **multi-level index**.
- Each index is rooted at an **inode** that stores the file's metadata and pointers to data.
  - Doesn't actually store file name or any data in the inode.
  - Inodes stores on a fixed block on disk in an **inode array**.
- Pointers are just block numbers.
  - **Direct pointers** point directly to the block (i.e. the pointer is a block number of a block that contains the file's data).
  - **Indirect pointer** points to an **indirect block** which is an array of direct pointers.
  - **Double indirect pointer** points to a **double indirect block** which is an array of indirect pointers.
  - Typically see up to triple indirect pointer.



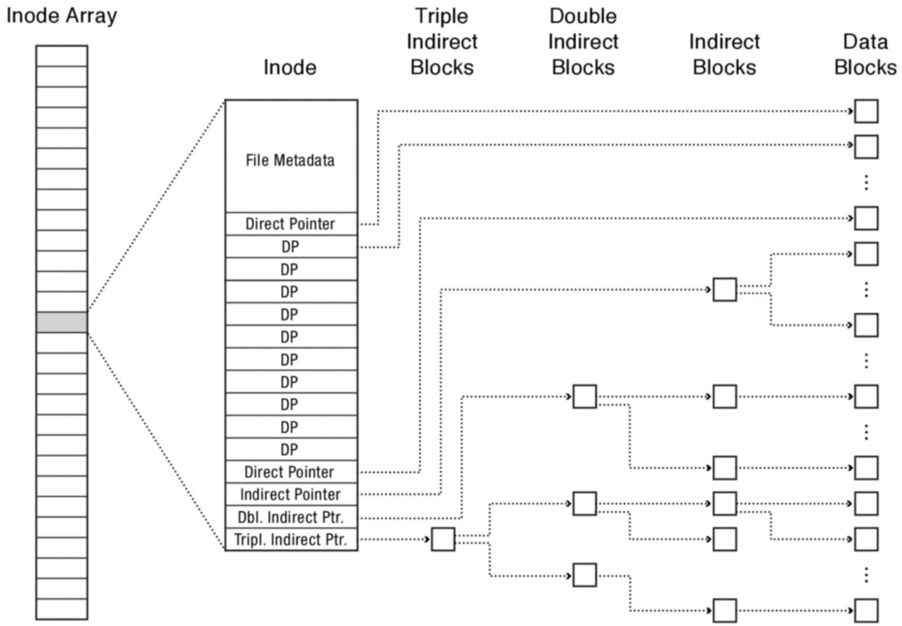
Fast File System (FFS)

Directory Structure

- Index of an inode in the inode array is the **inumber** which is used as the file number.
- Metadata stored in inode, not directory entry → hard links!

Free Space Management

- Bitmap where each bit corresponds to a block and indicates whether the block is free.
- Allocate disk blocks using **block group placement** which places file's inode block and data blocks near each other.
  - Reserve a fraction of disk space for optimal block group placement.



Concept Check

1. Consider an inode-based file system with 2 KiB blocks and 32-bit disk and file block pointers. Each inode has 12 direct pointers, an indirect pointer, a double indirect pointer, and a triple indirect pointer. How large of a disk can this file system support? What is the maximum file size?

Concept Check

1. Consider an inode-based file system with 2 KiB blocks and 32-bit disk and file block pointers. Each inode has 12 direct pointers, an indirect pointer, a double indirect pointer, and a triple indirect pointer. How large of a disk can this file system support? What is the maximum file size?

   32-bit disk $\rightarrow 2^{32}$ blocks $\rightarrow$ max disk size = 2^{32} blocks $\times$ 2^{11} bytes/block = 8 TiB

Concept Check

1. Consider an inode-based file system with 2 KiB blocks and 32-bit disk and file block pointers. Each inode has 12 direct pointers, an indirect pointer, a double indirect pointer, and a triple indirect pointer. How large of a disk can this file system support? What is the maximum file size?

   32-bit disk → 2^{32} blocks → max disk size = 2^{32} blocks × 2^{11} bytes/block = 8 TiB

   # pointers/block = 2^{11} bytes/block ÷ 4 bytes/pointer = 512 pointers per block

Concept Check

1. Consider an inode-based file system with 2 KiB blocks and 32-bit disk and file block pointers. Each inode has 12 direct pointers, an indirect pointer, a double indirect pointer, and a triple indirect pointer. How large of a disk can this file system support? What is the maximum file size?

   32-bit disk → 2^{32} blocks → max disk size = 2^{32} blocks ⨯ 2^{11} bytes/block = 8 TiB

   # pointers/block = 2^{11} bytes/block ÷ 4 bytes/pointer = 512 pointers per block

   Max file size = block size ⨯ number of all direct pointers (including ones from indirect pointers)

   = 2048 ⨯ (12 + 512 + 512^2 + 512^3)

   = 24 KiB + 513 MiB + **256 GiB**

Concept Check

2. Compare bitmap-based allocation of blocks on disk with a free block list.

3. For inode-based file systems, what is the point of having direct pointers? Why not just have indirect pointers since they can point to much more data than a single direct pointer?

Concept Check

2. Compare bitmap-based allocation of blocks on disk with a free block list.

   Bitmap = fixed size proportional to size of disk → better for contiguous allocation

   Free block list = shrinks as disk space is used up → better when disk utilization is high

3. For inode-based file systems, what is the point of having direct pointers? Why not just have indirect pointers since they can point to much more data than a single direct pointer?

Concept Check

2. Compare bitmap-based allocation of blocks on disk with a free block list.

   Bitmap = fixed size proportional to size of disk → better for contiguous allocation

   Free block list = shrinks as disk space is used up → better when disk utilization is high

3. For inode-based file systems, what is the point of having direct pointers? Why not just have indirect pointers since they can point to much more data than a single direct pointer?

   Most files are small enough to be contained within direct pointers → no need to waste an extra block read with indirect pointers.

Concept Check

4. List the set of disk blocks that must be read into memory in order to read the file `/home/cs162/pintos.bean` in its entirety from a UNIX BSD 4.2 file system (10 direct pointers, an indirect pointer, a double indirect pointer, and a triple indirect pointer) on a magnetic disk with 1 KiB block size. The `/home/cs162/pintos.bean` file is 15,234 bytes. Assume that the directories in question all fit into a single disk block each and the inode array is resident in memory (i.e. don't count inode array disk accesses).

Concept Check

4. List the set of disk blocks that must be read into memory in order to read the file `/home/cs162/pintos.bean` in its entirety from a UNIX BSD 4.2 file system (10 direct pointers, an indirect pointer, a double indirect pointer, and a triple indirect pointer) on a magnetic disk with 1 KiB block size. The `/home/cs162/pintos.bean` file is 15,234 bytes. Assume that the directories in question all fit into a single disk block each and the inode array is resident in memory (i.e. don't count inode array disk accesses).

	Inode Array
0	
1	
2	/
3	/bin
4	/home
5	/opt
6	/usr
7	/bin/cat
8	/home/cs152
9	/home/cs162
10	/opt/spark
11	/home/cs189_rant.txt
12	/usr/bin
13	/home/cs162/kidney.bean
14	/home/cs189/hw1
15	/home/cs162/pintos.bean

Concept Check

4. List the set of disk blocks that must be read into memory in order to read the file `/home/cs162/pintos.bean` in its entirety from a UNIX BSD 4.2 file system (10 direct pointers, an indirect pointer, a double indirect pointer, and a triple indirect pointer) on a magnetic disk with 1 KiB block size. The `/home/cs162/pintos.bean` file is 15,234 bytes. Assume that the directories in question all fit into a single disk block each and the inode array is resident in memory (i.e. don't count inode array disk accesses).

1. Read in inode for root directory.

	Inode Array
0	
1	
2	/
3	/bin
4	/home
5	/opt
6	/usr
7	/bin/cat
8	/home/cs152
9	/home/cs162
10	/opt/spark
11	/home/cs189_rant.txt
12	/usr/bin
13	/home/cs162/kidney.bean
14	/home/cs189/hw1
15	/home/cs162/pintos.bean

Concept Check

4. List the set of disk blocks that must be read into memory in order to read the file `/home/cs162/pintos.bean` in its entirety from a UNIX BSD 4.2 file system (10 direct pointers, an indirect pointer, a double indirect pointer, and a triple indirect pointer) on a magnetic disk with 1 KiB block size. The `/home/cs162/pintos.bean` file is 15,234 bytes. Assume that the directories in question all fit into a single disk block each and the inode array is resident in memory (i.e. don't count inode array disk accesses).

1. Read in inode for root directory.

	Inode Array
0	
1	
2	/
3	/bin
4	/home
5	/opt
6	/usr
7	/bin/cat
8	/home/cs152
9	/home/cs162
10	/opt/spark
11	/home/cs189_rant.txt
12	/usr/bin
13	/home/cs162/kidney.bean
14	/home/cs189/hw1
15	/home/cs162/pintos.bean

	Inode
DP	17
DP	
DP	
DP	
DP	
DP	
DP	
DP	
DP	
DP	
IP	
DIP	
TIP	

Concept Check

4. List the set of disk blocks that must be read into memory in order to read the file `/home/cs162/pintos.bean` in its entirety from a UNIX BSD 4.2 file system (10 direct pointers, an indirect pointer, a double indirect pointer, and a triple indirect pointer) on a magnetic disk with 1 KiB block size. The `/home/cs162/pintos.bean` file is 15,234 bytes. Assume that the directories in question all fit into a single disk block each and the inode array is resident in memory (i.e. don't count inode array disk accesses).

2. Read in block pointed to by the the first direct pointer for root inode.

	Inode Array
0	
1	
2	/
3	/bin
4	/home
5	/opt
6	/usr
7	/bin/cat
8	/home/cs152
9	/home/cs162
10	/opt/spark
11	/home/cs189_rant.txt
12	/usr/bin
13	/home/cs162/kidney.bean
14	/home/cs189/hw1
15	/home/cs162/pintos.bean

	Inode
DP	17
DP	
DP	
DP	
DP	
DP	
DP	
DP	
DP	
DP	
IP	
DIP	
TIP	

Name	File Number
bin	3
home	4
opt	5
usr	6

Concept Check

4. List the set of disk blocks that must be read into memory in order to read the file `/home/cs162/pintos.bean` in its entirety from a UNIX BSD 4.2 file system (10 direct pointers, an indirect pointer, a double indirect pointer, and a triple indirect pointer) on a magnetic disk with 1 KiB block size. The `/home/cs162/pintos.bean` file is 15,234 bytes. Assume that the directories in question all fit into a single disk block each and the inode array is resident in memory (i.e. don't count inode array disk accesses).

3. Read in block pointed to by the the first direct pointer for home inode.

	Inode Array
0	
1	
2	/
3	/bin
4	/home
5	/opt
6	/usr
7	/bin/cat
8	/home/cs152
9	/home/cs162
10	/opt/spark
11	/home/cs189_rant.txt
12	/usr/bin
13	/home/cs162/kidney.bean
14	/home/cs189/hw1
15	/home/cs162/pintos.bean

	Inode
DP	17
DP	
DP	
DP	
DP	
DP	
DP	
DP	
DP	
DP	
IP	
DIP	
TIP	

Name	File Number
bin	3
home	4
opt	5
usr	6

Concept Check

4. List the set of disk blocks that must be read into memory in order to read the file `/home/cs162/pintos.bean` in its entirety from a UNIX BSD 4.2 file system (10 direct pointers, an indirect pointer, a double indirect pointer, and a triple indirect pointer) on a magnetic disk with 1 KiB block size. The `/home/cs162/pintos.bean` file is 15,234 bytes. Assume that the directories in question all fit into a single disk block each and the inode array is resident in memory (i.e. don't count inode array disk accesses).

3. Read in inode for `home` directory.

	Inode Array
0	
1	
2	/
3	/bin
4	/home
5	/opt
6	/usr
7	/bin/cat
8	/home/cs152
9	/home/cs162
10	/opt/spark
11	/home/cs189_rant.txt
12	/usr/bin
13	/home/cs162/kidney.bean
14	/home/cs189/hw1
15	/home/cs162/pintos.bean

	Inode
DP	28
DP	
DP	
DP	
DP	
DP	
DP	
DP	
DP	
DP	
IP	
DIP	
TIP	

Concept Check

4. List the set of disk blocks that must be read into memory in order to read the file `/home/cs162/pintos.bean` in its entirety from a UNIX BSD 4.2 file system (10 direct pointers, an indirect pointer, a double indirect pointer, and a triple indirect pointer) on a magnetic disk with 1 KiB block size. The `/home/cs162/pintos.bean` file is 15,234 bytes. Assume that the directories in question all fit into a single disk block each and the inode array is resident in memory (i.e. don't count inode array disk accesses).

4. Read in block pointed to by the the first direct pointer for `home` inode.

	Inode Array
0	
1	
2	/
3	/bin
4	/home
5	/opt
6	/usr
7	/bin/cat
8	/home/cs152
9	/home/cs162
10	/opt/spark
11	/home/cs189_rant.txt
12	/usr/bin
13	/home/cs162/kidney.bean
14	/home/cs189/hw1
15	/home/cs162/pintos.bean

	Inode
DP	28
DP	
DP	
DP	
DP	
DP	
DP	
DP	
DP	
DP	
IP	
DIP	
TIP	

Name	File Number
cs152	8
cs162	9
cs189_rant.txt	11

Concept Check

4. List the set of disk blocks that must be read into memory in order to read the file `/home/cs162/pintos.bean` in its entirety from a UNIX BSD 4.2 file system (10 direct pointers, an indirect pointer, a double indirect pointer, and a triple indirect pointer) on a magnetic disk with 1 KiB block size. The `/home/cs162/pintos.bean` file is 15,234 bytes. Assume that the directories in question all fit into a single disk block each and the inode array is resident in memory (i.e. don't count inode array disk accesses).

5. Read in inode for `cs162` directory.

	Inode Array
0	
1	
2	/
3	/bin
4	/home
5	/opt
6	/usr
7	/bin/cat
8	/home/cs152
9	/home/cs162
10	/opt/spark
11	/home/cs189_rant.txt
12	/usr/bin
13	/home/cs162/kidney.bean
14	/home/cs189/hw1
15	/home/cs162/pintos.bean

	Inode
DP	28
DP	
DP	
DP	
DP	
DP	
DP	
DP	
DP	
DP	
IP	
DIP	
TIP	

Name	File Number
cs152	8
cs162	9
cs189_rant.txt	11

Concept Check

4. List the set of disk blocks that must be read into memory in order to read the file `/home/cs162/pintos.bean` in its entirety from a UNIX BSD 4.2 file system (10 direct pointers, an indirect pointer, a double indirect pointer, and a triple indirect pointer) on a magnetic disk with 1 KiB block size. The `/home/cs162/pintos.bean` file is 15,234 bytes. Assume that the directories in question all fit into a single disk block each and the inode array is resident in memory (i.e. don't count inode array disk accesses).

5. Read in inode for `cs162` directory.

	Inode Array
0	
1	
2	/
3	/bin
4	/home
5	/opt
6	/usr
7	/bin/cat
8	/home/cs152
9	/home/cs162
10	/opt/spark
11	/home/cs189_rant.txt
12	/usr/bin
13	/home/cs162/kidney.bean
14	/home/cs189/hw1
15	/home/cs162/pintos.bean

	Inode
DP	68
DP	
DP	
DP	
DP	
DP	
DP	
DP	
DP	
DP	
IP	
DIP	
TIP	

Concept Check

4. List the set of disk blocks that must be read into memory in order to read the file `/home/cs162/pintos.bean` in its entirety from a UNIX BSD 4.2 file system (10 direct pointers, an indirect pointer, a double indirect pointer, and a triple indirect pointer) on a magnetic disk with 1 KiB block size. The `/home/cs162/pintos.bean` file is 15,234 bytes. Assume that the directories in question all fit into a single disk block each and the inode array is resident in memory (i.e. don't count inode array disk accesses).

6. Read in block pointed to by the the first direct pointer for `/home/cs162` inode.

	Inode Array
0	
1	
2	/
3	/bin
4	/home
5	/opt
6	/usr
7	/bin/cat
8	/home/cs152
9	/home/cs162
10	/opt/spark
11	/home/cs189_rant.txt
12	/usr/bin
13	/home/cs162/kidney.bean
14	/home/cs189/hw1
15	/home/cs162/pintos.bean

	Inode
DP	68
DP	
DP	
DP	
DP	
DP	
DP	
DP	
DP	
DP	
IP	
DIP	
TIP	

Name	File Number
kidney.bean	13
pintos.bean	15

Concept Check

4. List the set of disk blocks that must be read into memory in order to read the file `/home/cs162/pintos.bean` in its entirety from a UNIX BSD 4.2 file system (10 direct pointers, an indirect pointer, a double indirect pointer, and a triple indirect pointer) on a magnetic disk with 1 KiB block size. The `/home/cs162/pintos.bean` file is 15,234 bytes. Assume that the directories in question all fit into a single disk block each and the inode array is resident in memory (i.e. don't count inode array disk accesses).

7. Read in inode for `/home/cs162/pintos.bean` file.

	Inode Array
0	
1	
2	/
3	/bin
4	/home
5	/opt
6	/usr
7	/bin/cat
8	/home/cs152
9	/home/cs162
10	/opt/spark
11	/home/cs189_rant.txt
12	/usr/bin
13	/home/cs162/kidney.bean
14	/home/cs189/hw1
15	/home/cs162/pintos.bean

	Inode
DP	68
DP	
DP	
DP	
DP	
DP	
DP	
DP	
DP	
DP	
IP	
DIP	
TIP	

Name	File Number
kidney.bean	13
pintos.bean	15

Concept Check

4. List the set of disk blocks that must be read into memory in order to read the file `/home/cs162/pintos.bean` in its entirety from a UNIX BSD 4.2 file system (10 direct pointers, an indirect pointer, a double indirect pointer, and a triple indirect pointer) on a magnetic disk with 1 KiB block size. The `/home/cs162/pintos.bean` file is 15,234 bytes. Assume that the directories in question all fit into a single disk block each and the inode array is resident in memory (i.e. don't count inode array disk accesses).

7. Read in inode for `/home/cs162/pintos.bean` file.

	Inode Array
0	
1	
2	/
3	/bin
4	/home
5	/opt
6	/usr
7	/bin/cat
8	/home/cs152
9	/home/cs162
10	/opt/spark
11	/home/cs189_rant.txt
12	/usr/bin
13	/home/cs162/kidney.bean
14	/home/cs189/hw1
15	/home/cs162/pintos.bean

	Inode
DP	280
DP	281
DP	282
DP	290
DP	291
DP	292
DP	293
DP	294
DP	295
DP	300
IP	305
DIP	
TIP	

Concept Check

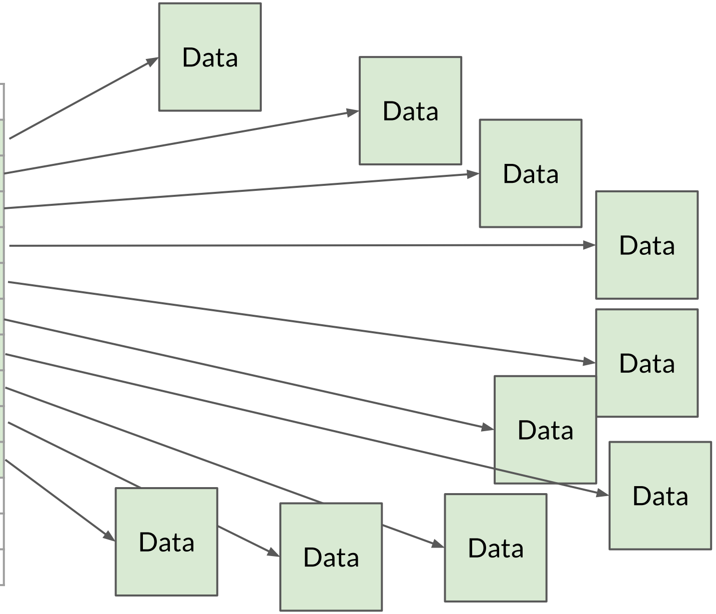
4. List the set of disk blocks that must be read into memory in order to read the file `/home/cs162/pintos.bean` in its entirety from a UNIX BSD 4.2 file system (10 direct pointers, an indirect pointer, a double indirect pointer, and a triple indirect pointer) on a magnetic disk with 1 KiB block size. The `/home/cs162/pintos.bean` file is 15,234 bytes. Assume that the directories in question all fit into a single disk block each and the inode array is resident in memory (i.e. don't count inode array disk accesses).

8-17. Read in each block pointed to by each direct pointer for `/home/cs162/pintos.bean` inode.

Bytes read = 1024 x 10 = 10240

	Inode Array
0	
1	
2	/
3	/bin
4	/home
5	/opt
6	/usr
7	/bin/cat
8	/home/cs152
9	/home/cs162
10	/opt/spark
11	/home/cs189_rant.txt
12	/usr/bin
13	/home/cs162/kidney.bean
14	/home/cs189/hw1
15	/home/cs162/pintos.bean

	Inode
DP	280
DP	281
DP	282
DP	290
DP	291
DP	292
DP	293
DP	294
DP	295
DP	300
IP	305
DIP	
TIP	



Concept Check

4. List the set of disk blocks that must be read into memory in order to read the file `/home/cs162/pintos.bean` in its entirety from a UNIX BSD 4.2 file system (10 direct pointers, an indirect pointer, a double indirect pointer, and a triple indirect pointer) on a magnetic disk with 1 KiB block size. The `/home/cs162/pintos.bean` file is 15,234 bytes. Assume that the directories in question all fit into a single disk block each and the inode array is resident in memory (i.e. don't count inode array disk accesses).

18. Read in indirect block from the indirect pointer in `/home/cs162/pintos.bean` inode.

Bytes read = 1024 x 10 = 10240

	Inode Array
0	
1	
2	/
3	/bin
4	/home
5	/opt
6	/usr
7	/bin/cat
8	/home/cs152
9	/home/cs162
10	/opt/spark
11	/home/cs189_rant.txt
12	/usr/bin
13	/home/cs162/kidney.bean
14	/home/cs189/hw1
15	/home/cs162/pintos.bean

	Inode
DP	280
DP	281
DP	282
DP	290
DP	291
DP	292
DP	293
DP	294
DP	295
DP	300
IP	305
DIP	
TIP	

Indirect Block
306
307
510
511
512

Concept Check

4. List the set of disk blocks that must be read into memory in order to read the file `/home/cs162/pintos.bean` in its entirety from a UNIX BSD 4.2 file system (10 direct pointers, an indirect pointer, a double indirect pointer, and a triple indirect pointer) on a magnetic disk with 1 KiB block size. The `/home/cs162/pintos.bean` file is 15,234 bytes. Assume that the directories in question all fit into a single disk block each and the inode array is resident in memory (i.e. don't count inode array disk accesses).

19-23. Read in direct blocks corresponding to the first five direct pointers in the indirect block. Last block is partially full.

Bytes read = 1024 x 10 + 1024 x 5 = 10240 + 5120 = 15,360

	Inode Array
0	
1	
2	/
3	/bin
4	/home
5	/opt
6	/usr
7	/bin/cat
8	/home/cs152
9	/home/cs162
10	/opt/spark
11	/home/cs189_rant.txt
12	/usr/bin
13	/home/cs162/kidney.bean
14	/home/cs189/hw1
15	/home/cs162/pintos.bean

	Inode
DP	280
DP	281
DP	282
DP	290
DP	291
DP	292
DP	293
DP	294
DP	295
DP	300
IP	305
DIP	
TIP	

Indirect Block
306
307
510
511
512