OS Homework — Chapter 1

Problem #10

Kernel mode vs. User Mode

#16 Block to write to disk — Only Block if disk buffer is full — otherwise it will write to buffer and not block.

#18 To save process state.
Yes, if it's multitasking (page 39)

#23 [1, 5, 9, 2] (p. 51 + 57) Remember @ 0.

#25 Block special — Random Access like disks
Character special — Stream data (p 44)

#27 Virtual Memory:
1. Save to disk if out of RAM
2. Each machine has own Address Space.

P174 #3 Need very low-level access to hardware.

Homework - Chapter 2

Homework

#5 All busy $(\frac{1}{2})^5 = \frac{1}{32} = 3.125\%$

#6 4GB - $\frac{1}{2}$GB = 3.5GB = 14x256MB processes.

need $x^{14} < .01 \Rightarrow x < .72 \Rightarrow 72\%$

#7 a) 80 min. b) $\frac{3}{4} \cdot x = 40 \Rightarrow x = \frac{160}{3} \approx 53\frac{1}{3}$ min

#12 Kernel Threads, otherwise the CPU would be idle during Disk I/O.

#14 Registers are part of a thread's state

#17 Single: Time = $12 + \frac{1}{3}75 = 37$ msec

requests = 27 / sec

Multi: 1000/12 = 83/sec

#18 Adv: custom thread scheduling

dis: whole process blocks

#22 Yes. Scheduler can wake up, monitor, and schedule.

Chapter 2 #26 No

#40 It would get more CPU time - good for more important processes

#44 3, 5, 6, 9, X ; if x > 9

#45 b) 6, 8, 10, 2, 4

$\frac{1}{5}(6+14+24+26+30) = 20$

c) 10, 6, 2, 4, 8 | $\frac{1}{5}(10+16+18+22+30) = 19\frac{1}{5}$

d) 2, 4, 6, 8, 10 | $\frac{1}{5}(2+6+12+20+30) = 14$

#49 40, 20, 40, 15

40

$\frac{1}{2}(40+20) = 30$

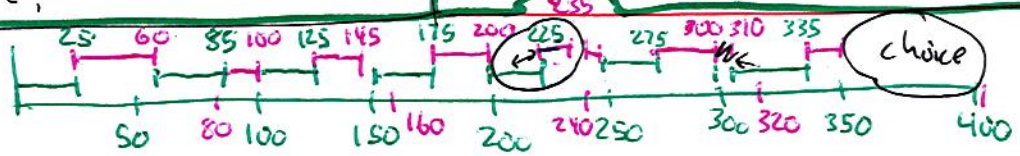
$\frac{1}{2}(30+40) = 35$

$\frac{1}{2}(35+15) = 25$

25

Homework

1.



2. Busy Waiting

3.

Producer

```
x = Produce
down(spaceSem)
putInQueue(x)
up(itemSem)
```

Consumer

```
down(itemSem)
x = getItemFromQueue();
up(spaceSem)
consume(x)
```

Schedule two processes using Earliest Deadline First

Process 1: P = 50, time = 25

Process 2: P = 80, time = 35

Solution

Homework | Chapter 3

#6 4k pages: (4, 3616), (8, 0), (14, 2656)

8k pages: (2, 3616), (4, 0), (7, 2656)

#7 20 → 8192 + 28 = 8212

4100 → ~~8292 8192~~ = 4100

8300 = 2·4096 + 108 → 6·4096 + 108 = 24684

#14 32 bits, 13 bits = page ⇒ 2^{19} pages = 2^{21} bytes → 2^{19} × 100 nsec = $2^{19} \times 10^{-7}$ sec

$= \frac{1}{2} \times 2^{20} \times 10^{-7}$ sec $\approx \frac{1}{2} \times 10^{6} \times 10^{-7} = \frac{1}{20}$ sec = 50 msec. = ½ = **50% time**

#15a $2^{48} \div 2^{12} = 2^{36}$ = 64 billion

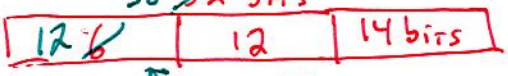
#22 $x \cdot 1 + (1-x) \cdot 5 = 2$ | $5 - 4x = 2$ | $3 = 4x$ | **$x = 75\%$**

#41 a) No - needs 17 pages. b) yes. Need 128, have 128

Homework Ch. 3 | #13 $\left(1 + \frac{n}{k}\right)$ n sec

#16 $1 \times (0.99) + 100 \times (.0099) + 6000000\,(.0001) = 601.98$ msec

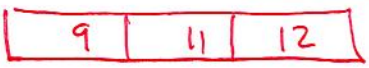
#17 a) Use less memory b)



one page-worth

Top level - 12 bits works well

#19



Pages are 2^{12} = 4K = 4096 bytes, 2^{20} pages

#24 $2^{48-13} = 2^{35} \approx$ 32 billion | #32 a) Yes, age > 400 b) No, age < 1000

#42 $n + 2000 \cdot 15000 = 60{,}000{,}000 \Rightarrow n = 30{,}000{,}000$

$30{,}000 + 2000 \cdot 7500 = 45{,}000{,}000$ = **45 sec**

Chapter 4 # 1, 6, 10, 25, 30, 35.

#1 ./etc/passwd

many other possible

./etc/./passwd

./etc/../etc/passwd

./././etc/passwd

./etc/../etc/passwd

#6 Yes, significant difference. Renaming merely changes data in a directory file. Copying and deleting uses (temporarily) a new inode entry, destroys links to the old file, and moves the file on disk.

#10 /usr/ast/x

#25 Including the two givens:

1000 0000 0000 0000
1111 1110 0000 0000

1111 1111 BBBB 0000
1000 000B BBBB 0000

1ccc cccB BBBB CC00
1ccc cccc 0000 CC00

Note: 'B' and 'C' are meant to stand for '1' bits, but are given to show the file in those blocks.

#30 For small files it would save disk space, as a 4k block would not need to be allocated. Also, if the inode is read into memory, as it is whenever a file is accessed, then if it is a small file, the whole file will be already present in the inode.

#35 (See pages 285-6)

f_1: 22, 19, 15, 17, 21,

f_2: 16, 23, 14, 18, 20

Homework | Chapter 5 #2 | yes. 802.11 is faster

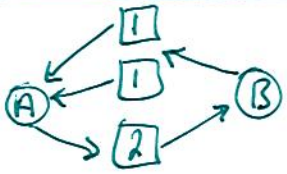
#13 Using kernel modules

#14
a) Device Drivers
b) Device Drivers / Interrupt handlers
c) Device Indep. OS Software
d) User-level

#16
- To share printer -
- Make it non-blocking

Chapter 6 #6, 8, 10, 14, 17, 21, 22, 26, 39

#6 "Don't block the box" attacks the hold + wait condition. A car cannot hold the intersection, while waiting for the intersection ahead to clear

#8 

In this setup, Process A acquired 2 copies of resource 1, (all available copies), then Process B acquired resource 2. Now Process A is blocked waiting for 2, and B is blocked waiting for any copy of 1.

#10 In neither case can deadlock occur. Since A already has all of its resources, it can not block, and so will eventually release R and S. Since processes B and C share only a single resource, no deadlock can occur between them.

#14 Only row 2 is $\leq A$. So we release process 2's resources, giving a new $A = (0,2,0,3,1)$. Now only row 3 is $\leq A$, so we release process 3's resources, giving $A = (0,2,0,3,2)$. At this point we see that processes 1 and 4 are deadlocked.

#17 Parallel processing

#22 No deadlock possible. If no resource is available now, then some process must be holding 2 copies, which is its max, so it will not block before releasing at least 1 resource. Then a waiting process can make progress

#21 If D gets 1, we are deadlocked. If C gets 1, we are fine.

#26 None at all. There is not enough of resource 5 to satisfy A, ever.

#39 Won't be on final.

(Semaphores will be, but not a hard problem like this.)

Will send solution later.