The University of Texas at Austin
Department of Electrical and Computer Engineering

Real-Time Systems / Real-Time Operating Systems

EE445M/EE380L.12, Fall 2020

Final Exam

Date: December 12, 2020

UT EID: _______________

Printed Name: ____
Last, First

Your signature is your promise that you have not cheated and will not cheat on this exam, nor will you help others to cheat on this exam:

Signature: ____

Instructions:

- Open book, open notes and open web.
- No calculators or any electronic devices other than your laptop/PC (turn cell phones off).
- You are allowed to access any resource on the internet, but no electronic communication other than with instructors.
- Please be sure that your answers to all questions (and all supporting work that is required) are contained in the space (boxes) provided.
- For all questions, unless otherwise stated, find the most efficient (time, resources) solution.

Problem 1	10	
Problem 2	10	
Problem 3	20	
Problem 4	20	
Problem 5	15	
Problem 6	10	
Problem 7	15	
Total	100	

Problem 1 (10 points): Deadlock

Assume the following *OS_Wait_Or* implementation from the midterm solutions:

```
 Sema4Type* OS_Wait_Or(Sema4Type* semaA, Sema4Type* semaB) {
  Sema4Type *semaRes;
  DisableInterrupts();
  while ((*semaA) <= 0 && (*semaB) <= 0) {
    EnableInterrupts(); DisableInterrupts();
  }
  if ((*semaA) > 0) {
    semaRes = semaA;
  } else {
    semaRes = semaB;
  }
  (*semaRes) = (*semaRes) – 1;
   EnableInterrupts();
  return semaRes;
 }
```

Given the following program using *OS_Wait_Or*:

```
OS_InitSemaphore(&A,1);
OS_InitSemaphore(&B,1);
OS_InitSemaphore(&C,1);
OS_InitSemaphore(&D,1);
```

```
TaskA(){
 sema4* t1;
 sema4* t2;
 t1= OS_Wait_OR(&A,&B);
 t2= OS_Wait_OR(&B,&C);
 ...

 OS_Signal(t2);
 OS_Signal(t1);
}
```

```
TaskB(){
 sema4* t1;
 sema4* t2;
 t1= OS_Wait_OR(&A,&C);
 t2= OS_Wait_OR(&C,&D);
 ...

 OS_Signal(t2);
 OS_Signal(t1);
}
```

```
TaskC(){
 sema4* t1;
 sema4* t2;
 t1= OS_Wait_OR(&D,&A);
 t2= OS_Wait_OR(&B,&C);
 ...

 OS_Signal(t2);
 OS_Signal(t1);
}
```

Is there any possible interleaving that could cause deadlock with this program? If yes, please show an example that could be a deadlock. If not, justify your answer.

Problem 2 (10 points): Scheduling

Given the following three foreground threads in your system:

```
ThreadA(){
  while(1) {
    OS_Wait(&Sema);
    countA++;
  }
}
```

```
ThreadB(){
  while(1) {
    OS_Wait(&Sema);
    countB++;
  }
}
```

```
ThreadC() {
  while(1) {
    OS_Signal(&Sema);
    OS_Signal(&Sema);
    countC++;
  }
}
```

Is there any OS realization (scheduling strategy, semaphore implementation, priority assignment) that ensures that all three threads run an equal number of times, i.e. that the three counting variables *countA*, *countB* and *countC* are equal or at most off by one at all times? If so, what is that OS setup? If not, why not? You can assume that the semaphore and all counting variables are initialized to zero.

Problem 3 (20 points): Heap and ELF loader

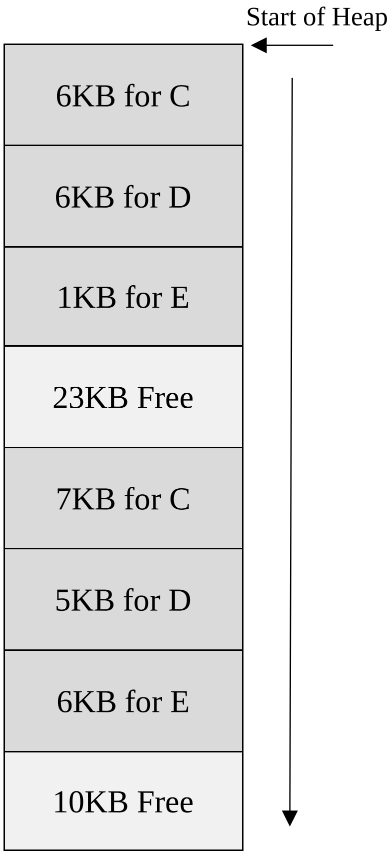
Assume that six ELF executables files with code and data segment sizes as shown below are stored on your SD card and launched from your interpreter in the order *A, B, C, D* and *E*, where the ELF loader uses different heap allocation routines for code and data. *Heap_Alloc* allocates memory in 1KB granularity from start to end, i.e. an allocated block is always placed at the bottom of its chosen free space. Available heap size is 64KB in total and there is no metadata overhead.

Size of code and data segment for each program:

Program	*Code size*	*Data size*
A	6 KB	11KB
B	6 KB	13KB
C	6 KB	7 KB
D	6 KB	5 KB
E	6 KB	1 KB

ELF loader:

```
int exec_elf(…) { int sr;
  …
  sr = StartCritical()
  code = Heap_AllocCode(h->codeSize);
  data = Heap_AllocData(h->dataSize);
  EndCritical(sr);
  …
}
```



Given the heap state as shown on the right:

a) What heap allocation algorithms (first/best/worst fit) are used for code and data? Are they different? How do you know, i.e. explain your reasoning.

b) What was the program execution order flow, i.e. specify the sequence and order in which programs started and ended to reach this heap state, e.g. in the form *Start A*, *End A*, …

c) Suppose you launch program *E* multiple times and there are no other programs loaded. How many instances of *E* can be loaded into memory simultaneously? Assume that there are enough resource other than the heap.

d) Now suppose you want to load 50 instances of program *E* into memory simultaneously. Is this possible? If no, why not. If yes, explain how to enable this with 64KB of heap.

Problem 4 (20 points): File Systems

In a filesystem, disk accesses are expensive and can degrade performance. Given a disk with 512 byte blocks and the following trace of file accesses made by an application:

a1: Read File A, Byte 634
a2: Write File A, Byte 634
a3: Read File B, Byte 128
a4: Read File A, Byte 42
a5: Write File B, Byte 128
a6: Read File C, Byte 522
a7: Write File C, Byte 1096
a8: Write File A, Byte 42

a) How many disk accesses are performed by file systems that use an indexed and linked allocation as discussed in class? Assume that the directory and index table are both already loaded into memory, but no other caching of data is performed between accesses.

Indexed File System:	Linked File System:

b) Now assume an implementation of a file system in which you have created a small file system cache in memory that remembers the 3 last blocks from the disk that you have accessed. Suppose that the cache replaces blocks in a First In First Out manner. How many disk accesses are performed for the indexed and linked file systems? Assume that modified (dirty) blocks in the cache are only written back to disk when they are evicted from the cache, and that linked list traversals are always started from the beginning.

Indexed File System:	Linked File System:

c) To have persistence in case of crashes, a periodic writeback of dirty blocks in the cache must be factored in. Assume that after the fifth access in the trace, we perform a periodic writeback of all dirty elements in the cache. How does this affect the number of disk accesses for the index and linked file systems?

Indexed File System:	Linked File System:

d) How does the write back frequency affect file system performance and reliability?

Problem 5 (15 points): Virtual Memory

Assume a computer with a 16-bit memory address space using virtual memory with 4kB pages. Given the two ELF executable files containing code and data segments with sizes and load addresses as indicated, the partial state of main memory, and the page table of the process executing program *A* after it has been loaded into memory. No other process is loaded at this point:

Program A:

	Load address	*Size*
Code	0x0000	6kB
Data	0x2000	12kB

Program B:

	Load address	*Size*
Code	0x0000	4kB
Data	0x6000	13kB

Page Table A	Page Table B	Memory	
4		OS Code	0x0000
5		OS Data	
14			
7			
9			
			0xFFFF

a) What is size of each page table per process?

b) Show the location of the code and data segments for process A in memory. Is there any internal or external fragmentation? What is the largest program (code and data size) that can be loaded?

c) Now assume that program *B* is also loaded into memory. Fill the page table for the corresponding process and show the updated memory state after *B* has been loaded. Assume that memory is allocated in a first available fashion from bottom to top. Is there any external or internal fragmentation now? What is the largest program that can now be loaded?

Problem 6 (10 points): Relocation

For each of the following functions written in assembly, is the code position-independent? If not, indicate which part of the code is not and what needs to be done to relocate it at load time.

a)

```
funcA
  LDR R1,=count
  LDR R0,[R1]
  ADD R0,R0,#1
  SVC #42
  BX  LR
```

b)

```
funcB
  ADD R1,R9,#32
  LDR R0,[R1]
  ADD R0,R0,#1
  BL  OS_Sleep
  BX  LR
```

Problem 7 (15 points): Networking

Assume that two computers are directly connected in a local Ethernet network running at a raw physical layer bit rate of 10Mbit/s.

a) What is the maximum achievable bandwidth for transmitting data from one machine to the other at the Ethernet link layer? Assume that no other machines are transmitting, i.e. there are no collisions.

b) Now assume that we are running a TCP/IP protocol over the local network. What is the maximum achievable bandwidth for transmitting data from one machine to the other at the IP layer? Again, assume that no other machines are transmitting, i.e. there are no collisions and that the IP protocol does not use any options.

c) Finally, what is the maximum achievable bandwidth for transmitting data from one machine to the other over a UDP and TCP connection? Assume that the TCP connection is already established and that there are no buffering, windowing or acknowledgment delays.