CS 367-001/003: **Homework 6** Mem. Mgmt / Processes & Threads

Due Monday 12/5 or Tuesday 12/6, on paper, at start of class. You may work with one partner.

Late penalty is 25% per day. You are not expected to edit this document; please write your solutions on paper or type up your solutions in a new file. Each question is worth ten points. For the first five, you'll likely write a couple sentences to a paragraph or two each.

We discussed immediate coalescing as well as only coalescing at allocation time (that is, deferred coalescing only when two adjacent frees are visited in order). We also discussed first fit, next fit, and best fit strategies. Consider these variations in the first few questions below.

1. If we are using first fit, compare immediate coalescing with allocation-time coalescing. How much effort (in terms of number of blocks coalesced) is needed for an individual coalesce action? Does either require more overall time?

2. Compare the three allocation strategies (first/next/best) by the time taken for an allocation in each of these situations. Mention anything else that seems relevant to their performance.
   a) entirely empty heap
   b) heap with a few contiguous allocations early on
   c) mostly-full heap with very large allocated blocks
   d) mostly-full heap with very small allocated blocks

3. Received signals aren't queued. Why does this matter? How specifically might this lead to a bug? How do we deal with the situation? Be as specific as you can.

4. What is assumed to be true about our stack when relating the location of a setjmp call and the location of a corresponding longjmp call? What is stored as part of a jmp_buf?

5. What is shared between threads? What is shared between processes? If you want two control flows to repeatedly interact and communicate with each other, which do you probably want to use? Justify your position with properties about ***both*** threads and processes.

For the problems below, you will be tempted to type in the program and run it to get your answers. This is fine for checking your answer but I strongly suggest you try to figure out the answer FIRST.

6. Consider the following C program. How many times will each string ("A", "B", "C", "D", "E") be output?

```
fun() {
  printf("A\n");
  fork();
  printf("B\n ");
  if (fork() != 0) {
     printf("C\n ");
     fork();
     printf("D\n ");
  }
  printf("E\n ");
}
```

7. (two parts!) (a)What are all of the possible output sequences of the following program? (b)If the `waitpid()` call is removed, what are all of the possible outputs?

```
int main() {
   if (fork() == 0) {
     printf("a\n");
     exit(0);
   }
   else {
     printf("b\n");
     waitpid(-1,null,0);
   }
   printf("c\n");
   exit(0);
}
```

8. What are all of the possible outputs of this program? (main on the right).

```c
int x = 0;
pid_t pid;
int flag=1;

void handler1(int sig) {
    x=x+1;
    printf("%d\n",x);
    fflush(stdout);
    exit(0);
}

void handler2(int sig) {
    x = x + 2;
    flag = 0;
    waitpid(-1, NULL, 0);
}
```

```c
int main() {
    signal(SIGUSR1, handler1);
    signal(SIGCHLD, handler2);

    printf("%d\n",x);
    fflush(stdout);

    if ((pid = fork()) == 0) {
        while(1) {};
    }
    kill(pid, SIGUSR1);
    while(flag) {};
    printf("%d\n",x);
    fflush(stdout);
    exit(0);
}
```

---

9. Consider the following C program.

```c
fun() {
  printf("A\n");
  if (fork()==0){
    printf("B\n");
    fork();
    printf("C\n");
    exit();
  }
  printf("D\n");
  if (fork() == 0) printf("E\n");
  printf("F\n");
}
```

Which of the following outputs are legal?
(Disregard the missing \n in the output)

a) ADBCEFCF
b) ABCDEFCF
c) ADCEFBCF
d) ABCECDFF
e) ADEFBCFC

---

10. Consider the following C program.

```c
B() {
   pid_t pid;
   if ((pid=fork())!= 0)
       waitpid(pid,NULL,0);
   printf("2\n");
   if (fork() == 0)
      { printf("3\n"); exit(0); }
   printf("5\n");
   exit(0);
}
```

Which of the following outputs are possible?
(Disregard the missing \n in the output)

a) 232553
b) 235325
c) 232355
d) 235253
e) 252533