CS 214 | Recitation Handout

#1 Code review - how would you improve this code. Highlight every error you notice and then discuss the worst ones

```
1    // A program to run many commands in parallel
2    // Lines that start with an ! are executed
3    int main(int argc, char** argv) {
4      if(argc!=2) { printf("Usage: %s commandfile\n", argv); exit(1); }
5      size_t capacity = 200;
6      char* buffer  = malloc(capacity);
7      ssize_t bytes;
8      FILE *file = fopen(argv[1],"r");
9      if(!file) { perror("Could not read file"); return 1;}
10     while( 1 ) {
11       bytes = getline(& buffer, & capacity , file );
12       buffer[bytes-1] = 0;
13       puts(buffer);
14       if( strcmp(buffer, "END") || bytes == -1) break;
15       if(*buffer == '!') {
16         if( ! fork() ) { execlp( "bash", buffer +1 , (char*) NULL); exit(1);}
17       }
18     }
19     return 0;
20   }
```

Line number : Comment or suggested fix

#2 What are POSIX signals?

#3 What are the two sources of signals?

#4 What are the most well known signals and what do they do?

SIGINT
SIGSEGV
SIGKILL

Demo.

First let's create an unsuspecting long running process ...

```
// dotwriter.c
int main() {
  printf("My pid is %d\n", getpid() );
  int i = 60;
  while(--i) {
    write(1, ".",1);
    sleep(1);
  }
  write(1, "Done!",5);
  return 0;
}
```

How can I send a signal from another program?

```
int main(int argc, char** argv) {
   int signal = atoi(argv[1]);
   pid_t pid = atoi( argv[2] );

  if(signal && pid) _______________
  return 0;
}
```

How can I send a signal from the terminal?

#5 How would you modify the dotwriter program to send itself a SIGINT, after 5 dots?

#6 Alarming signals

```
void main() {
   char result[20];
   puts("You have 4 seconds");
   while(1) {
      puts("Secret backdoor NSA Password?");
      char* rc = fgets( result, sizeof(result) , stdin);
      if(*result='#') break;
  }
  puts("Congratulations. Connecting to NSA ...");
  execlp("ssh", "ssh", "nsa-backdoor.net", (char*)NULL);
  perror("Do you not have ssh installed?"); return 1;
}
```

#7 Stopping and continuing programs

```
SIGSTOP
SIGCONT
```

#8 Shell demo Background processes and redirection (>) pipes (|)

```
&
ps
jobs
fg
bg
nohup
wc *.c > data.txt
```

#9 Spot the errors part 1

```
// Spot the errors part 1
double *a = malloc( sizeof(double*) );
double *b = a;
free(b); b = 0;
*a = (double) 0xbaadf00d;

char* result;
strcpy(result, "CrashMaybe");

void* append(char** ptr, const char*mesg) {
  if(!*ptr)  ptr = malloc( strlen(mesg) );
  strcat( *ptr, mesg);
}
```

#10 Spot the errors part 2

```
//Spot the errors part 2
char* f() {
char result[16];
strcat( result, "Hi");
int *a;
if( &a != NULL) { printf("Yes %d\n",42); }

struct link* first= malloc(sizeof(struct link*));
free(first)
if(first->next) free(first->next);
return result;
}
```