

## #1 Introducing exec family

## #2 A most powerful program. Can we fix it?

```

1: int main(int argc, char**argv) {
2:     printf("Executing %s ...\n", argv+1);
3:     execvp( argv + 1, argv + 1);
4:     perror("Failed to be all powerful");
5: }

```

## #3 Implement our own version of cat

\* Usage text

\* Potential errors?

```

1: int main(int argc, char**argv) {
2:     if(argc != 2)
3:         fprintf(stderr, "Usage: %s filename\n", argv[0]);
4:
5:     FILE* file = fopen(argv[1], "r"); // may return NULL
6:     char* line = NULL;
7:     size_t capacity;
8:     ssize_t bytesread;
9:     int linenumber = 0;
10:    while(1) {
11:        bytesread = getline( &line, &capacity, file);
12:        if(bytesread == -1) break;
13:        printf("%3d: %s", linenumber++, line);
14:    }
15:    free(line);
16:    fclose(file);
17:    return 0;
18: }

```

## #4 Puzzle: Fix my getline implementation. What asserts might you add?

```

1: ssize_t mygetline(char **lineptr, size_t *n, FILE *f){
2:     what asserts would you add here?
3:
4:     if( _____) { *n = 256; _____ = malloc(*n);}
5:     size_t bytesread = 0;
6:     int c = 0;
7:     while( ferror(f)==0 && feof(f)==0 ){
8:         if (bytesread == *n) { /* extend buffer */
9:
10:        }
11:        c = fgetc(f);
12:    }
13:    return -1; // error (e.g. end of file)
14: }

```

## #5 Fix me! What is wrong with the following?

```

1: int main(int argc, char** argv) {
2:     char** lineptr;
3:     size_t size;
4:     size = getline(lineptr, &size, stdin);
5:     execlp(lineptr);
6:     return 0;
7: }

```

## #6 Environmental Variables

- What is `getenv("HOME");`
  - What is `getenv("PATH")`
  - What is `getenv("USER");`
  - What is `getenv("AWESOME");`
  - `extern char** environ;`
- 

## #7 Puzzle What does the following example do? How does it work?

```
1: int main() {
2:     close(1); // close standard out
3:     open("log.txt", O_RDWR | O_CREAT | O_APPEND, S_IRUSR |
4:     S_IWUSR);
5:     puts("Captain's log");
6:     chdir("/bin");
7:     execl("/bin/ls", "/bin/ls", ".", (char*)NULL); // "ls ."
8:     perror("exec failed");
9:     return 0; // Not expected
10: }
11:
```

---

## #8 Puzzle My two processes can guess the same random number!?

```
1: int main(int argc, char**argv) {
2:     time_t seconds = time(NULL); // seconds since Jan 1, 1970
3:     srand( seconds ); // initialize random number generator
4:
5:     pid_t child = fork();
6:     printf("My fork value is %d\n", (int) child );
7:     int r = rand() & 0xf;
8:     printf("%d: My random number is %d\n", getpid(), r);
9:     return 0;
10: }
```