CS341 #22 - Pipes and file seeks

1 How do you use unnamed pipe to send a message from the parent to the child?

2. What is fseek and ftell? How would you use them?

3. What happens to the other process if you fclose after forking?

4. What happens to the other process if you fseek before forking?

5. What happens to the other process if you fseek after forking?

6. Why does pwrite exist? When would you use it?

7. What is an named pipe and an unnamed pipe?

8. What signals can a pipe generate and when?

9. How would you modify your pipe code to send an integer value of a variable?

10. Why is it necessary to close the pipe's unused filedescriptors after forking?

11. How would you fix/improve this code?

```
pthread_mutex_t m;
pthread_cond_t cv;
int in, out, count;
void* buffer[16]

void enqueue(void* ptr) {
  p_m_lock(&m);
  while(count < 16) {}
  pthread_mutex_unlock(&m);
  p_cond_broadcast(&cv);
  count ++;
  buffer[ (in++) % 16 ]  = ptr;
}
```

```
void* dequeue() {
  p_m_lock(&m);
  while(count  == 0) {}
  void* result = buffer[ (out++) % 16 ];
  p_cond_broadcast(&cv);
  pthread_mutex_unlock(&m);
  count --;
  return result;
}
```

```
void pipe_or_quit(int*result) {
  if( 0 == pipe(result) ) return; else quit("pipe");
}

void create_pipes(int* array6) {
  pipe_or_quit(array6);
  pipe_or_quit(array6 +2);
  pipe_or_quit(array6 +4);
}


void exec_or_quit(const char *program, const
char **args, int old_err_fd) {
  execv(program, (char*const*) args);
  dup2(old_err_fd, 2);
  quit("execv");
}
```

```
int run(const char *test, const char *prog, const char
**args, const char *input, char **output,
          char **erroroutput, int *waitresult) {
  if (test) printf("%s: Running %s\n", test, prog);
  int pipes[6];
  create_pipes(pipes);
  pid_t childid = fork_or_quit();
  if(childid ==0 ) {
//Child should close 'in'(input), out(output) err(output)
   // close unused end of pipes
   close(pipes[1]); close(pipes[2]);close(pipes[4]);
   int old_err_fd = dup(2);
   dup2_or_quit(pipes[0] /*read from */,0);
   dup2_or_quit(pipes[3] /*write to*/,  1);
   dup2_or_quit(pipes[5] /*write to*/  ,2);
   alarm(ALARM_TIMEOUT_SECONDS);
   exec_or_quit(prog, args, old_err_fd);
  }
```