|  |  |
| --- | --- |
| **CS 341 #22** | ***Pipes and Seeks*** |

**> Warm up**

Can you rewrite the following ...

|  |  |
| --- | --- |
| **1:**  **2: 3:** | **char mesg[100]; sprintf("hello %d\n", 123); write(1, mesg, strlen(mesg) );** |

using asprintf?

using dprintf?

**>** Remember this for later: **dup2(**existingfd**,** newfd**)**

|  |  |
| --- | --- |
| **1:**  **2: 3:** | **int fd = open("log.txt", ...,...) dup2(fd, 1); write/fork/exec** |

**>** Remember this for later: **fdopen(**fd, ....**)**

|  |  |
| --- | --- |
| **1:**  **2: 3:** | **int fd = open("log.txt", ...,...) FILE\* f = fdopen(fd,"w") fprintf(f, "hello!");** |

**> Pipes! (demo)**

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

**> Random access**

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?

**> Pipes (part 2)**

7. What is a named 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 useful to close a pipe's unused filedescriptors after forking?

> **Code Review: Can you improve this queue code?**

|  |  |
| --- | --- |
| int in, out, count;  void\* buffer[16]  void enqueue(void\* ptr) {  pthread\_mutex\_lock (&m);  while(count < 16) {**/\*loop\*/**}  pthread\_mutex\_unlock(&m);  p\_cond\_broadcast(&cv);  count ++;  buffer[(in++)**%16** ] = ptr;  } | void\* dequeue() {  pthread\_mutex\_lock(&m);  while(count == 0) {**/\*loop\*/**}  void\* res = buffer[(out++)**%16**];  p\_cond\_broadcast(&cv);  pthread\_mutex\_unlock(&m);  count --;  return res;  } |

**> Altogether now... Build Descarte's Demon AKA an *autograder*!**

|  |  |
| --- | --- |
| **1: 2:**  **3: 4:**  **5: 6:**  **7: 8:**  **9:**  **10:**  **11:**  **12:**  **13: 14: 15: 16:** | int p[6];  pipe(p);  pipe(p + \_\_\_ )  pipe(p + \_\_\_ )  pid\_t childid = fork();  if( childid ==0 ) {   dup2 (p[0] /\*read from \*/, \_\_\_);  dup2 (p[3] /\*write to\*/, \_\_\_);  dup2 (p[5] /\*write to\*/ ,\_\_\_);   //Child should close 'in'(input), out(output) err(output)  close(p[1]) close(p[2]);close(p[4]);  alarm(10) // Max 10 seconds for test to run;  execlp(prog, prog, NULL)  } |