|  |  |
| --- | --- |
| **CS 241** | **Lecture Handout #5** |

**#1 Introducing exec**

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

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

**#3 Implementing a our own version of cat  
\* Usage text  
\* Potential errors?**

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

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

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

**Puzzle #3 Fix me!** What is wrong with the following?

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

**Environmental Variables**

* What is getenv("HOME");
* What is getenv("PATH");
* What is getenv("USER") ;
* What is getenv("AWESOME");
* char\*\* environ

**Puzzle #4** What does the following example do? How does it work?

|  |  |
| --- | --- |
| **1: 2:**  **3: 4:**  **5: 6:**  **7: 8:**  **9:**  **10:**  **11:** | int main() {  close(1); // close standard out  open("log.txt", O\_RDWR | O\_CREAT | O\_APPEND, S\_IRUSR | S\_IWUSR);  puts("Captain's log");  chdir("/usr/include");  execl("/bin/ls", "/bin/ls",".",(char\*)NULL); // "ls ."  perror("exec failed");  return 0; // Not expected  } |

**Puzzle #5 Why is this 'broken'?**

|  |  |
| --- | --- |
| **1: 2:**  **3: 4:**  **5: 6:**  **7:**  **8:**  **9:** | int main(int argc, char\*\*argv) {  srand(time(NULL));  pid\_t child = fork();  printf("My fork value is %d\n", (int) child );  int r = rand() & 0xf;  printf("%d: My random number is %d\n", getpid(), r);  return 0;  } |

**Puzzle #6 What is this madness :-)**

|  |  |
| --- | --- |
| **1: 2:**  **3: 4:**  **5: 6:**  **7:**  **8:**  **9:** | int main(int c, char \*\*v)  {  while (--c > 1 && !fork());  int val = atoi(v[c]);  sleep(val);  printf("%d\n", val);  return 0;  } |

If there's time for more snow...

|  |  |
| --- | --- |
| **1: 2:**  **3: 4:**  **5: 6:**  **7: 8:**  **9:**  **10:**  **11:**  **12:**  **13:**  **14:** | **void snowflake() {**  **srand((unsigned)time(NULL));**  **int col = rand() % cols;**  **int row = 0;**  **while (row < rows) {**  **gotoxy(row, col);**  **fprintf(stderr, "\*");**  **usleep(200000);**  **gotoxy(row, col);**  **fprintf(stderr, " ");**  **row++;**  **}**  **}** |