|  |  |
| --- | --- |
| **CS 241 #3** | ***A day at the C Side*** |

**Puzzle 1: What is the value of** p[strlen(p)] **, if** p **points to "Sys" ?**

**Hint: strlen("") ==0 strlen("@")==1, strlen(NULL)==**#@?#WT?!

**Q1: How do I find out how to use  
 \_\_\_< useful function or system call here>\_\_?**

|  |  |
| --- | --- |
| **$** |  |

**Puzzle 2: How do I find out how to use stat in C?**

**Q2: What are the manual sections?**

* Section 2:

* Section 3:
* Section 7:

**Q3: How do I allocate and free heap memory in C?**

* Allocate:

* Free:

**Q4: Can I make a pointer *really free* by freeing it twice?**

**Q5: What do we call a pointer that has been free’d?**

**Best Practice:** Always set free’d pointers to NULL.

|  |  |
| --- | --- |
| **1:**  **2:**  **3:** | **// ... code ...**  **free(ptr);**  **ptr = 0;** |

**Puzzle 3: Fix a custom string concatentation function:**

|  |  |
| --- | --- |
| **1: 2:**  **3: 4:**  **5: 6:**  **7: 8:**  **9:**  **10:**  **11:**  **12:**  **13:**  **14:** | **void mystrcat(char \*dest, const char \*src) {**  **while (\*src) {**  **dest = src;**  **src++; dest++;**  **} }** |

|  |  |
| --- | --- |
| Address | Memory Contents |
| 1000 | '!' |
| 1001 | '2' |
| 1002 | 'B' |
| 1003 | '\0' |
| ... | |
| 2000 | '2' |
| 2001 | 'B' |
| 2002 | '|' |
| 2003 | '\0' |
| 2004 |  |
| 2005 |  |
| 2006 |  |
| ... | |

***Puzzle 4 - Walk Through***

|  |  |  |
| --- | --- | --- |
| **Type** | **Variable** | **Memory Addr.** |
| **const char \*** | **src** | **0x1000** |
| **char \*** | **dest** | **0x2000** |

* **Line 3:** What does **(\*src)** do?
* **Line 4:** What does **(dest = src)** do?
* **Line 3..9:** When does the loop exit?

**Puzzle 5: Fix my custom string duplication function**

|  |  |
| --- | --- |
| **1: 2:**  **3: 4:**  **5: 6:**  **7: 8:**  **9:**  **10:**  **11:** | **char \*mystrdup(const char \*src) {**  **char \*p = sizeof(src);**  **strcpy(src, p);**  **return p; }** |

**Q6: What is the purpose of a file stream, just files?**

A “file stream” (or “file descriptor” in system calls) is the base interface to EVERYTHING external to RAM. This includes:

* Standard Streams:  
    
  + **stdin**:

* + **stdout**:
  + **stderr**:

**Q7: Writing to file streams: fprintf**

What if the output of the following code snippet?

|  |  |
| --- | --- |
| **1: 2:**  **3: 4:** | **fprintf(stderr, "CS 241: ");**  **fprintf(stdout, "System ");**  **fprintf(stderr, "Programming ");**  **fprintf(stdout, "\n");** |

* Result:

**Q8: What is asprintf()?**

|  |
| --- |
| **int asprintf(char \*\*strp, const char \*fmt, ...)** |

* **char \*\*strp:**
* **const char \*fmt:**

**Puzzle 6: Pointer Arithmetic**

|  |  |
| --- | --- |
| **1: 2:**  **3: 4:**  **5: 6:**  **7:**  **8:**  **9:**  **10:**  **11:** | **// Count the number of elements in an int-array**  **// before the number -1 appears in the array:**  **int count\_before(int \*array) {**  **int \*ptr = array;**  **while (\*ptr != -1) { ptr++; }**  **return (ptr – array) / \_\_\_\_\_\_\_\_\_\_\_\_;**  **}** |

**Debug Less: Use assert e.g. assert(ptr && counter > 5);**

C provides the library macro assert that be used to find bugs in debugging and completely disappear in production code! Two modes:

* Debug mode (-g flag):
* Production mode (#NDEBUG):

|  |
| --- |
| **Best Practice:** Always assert pre-conditions and assumptions. |

**Puzzle 7: Putting it altogether**

|  |  |
| --- | --- |
| **1: 2:**  **3: 4:**  **5: 6:**  **7: 8:**  **9:**  **10:**  **11:**  **12:**  **13:**  **14:**  **15:**  **16:**  **17:**  **18:**  **19:**  **20:**  **21:** | **// Sum an array of positive numbers, storing**  **// the result in `result` (by ref) // and use asprintf to return a text version of result**  **char\* mysum(const int \*ptr, int \*result) {**    **while ( \*ptr ) {    sum += \*(ptr++);    }**  **char \*text = NULL;**    **asprintf(**  **return text;**  **}** |