Parsing in C

- Functions available
  - char *strtok(char *str, const char *delim);
    - tokenize a string on any of a number of delimiters
    - **alters the string!**
  - char *strstr(const char *haystack, const char *needle);
    - find a substring, return pointer to substring or NULL
  - char *strchr(const char *s, int c);
    - find the first occurrence of a character, return pointer to c or NULL
  - char *strrchr(const char *s, int c);
    - find the last occurrence of a character, return pointer to c or NULL
  - int isdigit(int c)
    - is the char a digit? return 0 if false
    - many is*type*() functions exist

strok_r

- strtok_r is great for *tokenizing* a string start to finish with a variable set of delimiters
  - **ALTERS THE STRING**

- **`pToken = strtok_r(pBuffer, " :", &pSavePtr);`**
  - find the first token delimited by space or :

- **`pToken = strtok_r(NULL, " :", &pSavePtr);`**
  - find the second token delimited by space or :

- Does not work for MathPacket!

Notes

- #include <string.h>
- #include <ctype.h>

Math Packet

- What does a MathPacket look like?

```
MATH/1.1 100 OK\n
Result: 2\n
Rounding: False\n
X-Server-Version: 1.1.4\n
Overflow: False\n
Connection: Close\n
\n
```

```
MATH/1.1 100 OK\nResult: 2\nRounding: False\nX-Server-Version: 1.1.4\nOverflow: False\nConnection: Close\n\n
```

Parsing

- Zero out the buffer!
  - memset(pBuffer, ‘\0’, SIZE)
- recv() until you find \n\n
- Search for a header **from the start** of the packet
  - temporarily alter the string to help you
  - leave the string unaltered after you are done
  - repeat for the next header.

Oh no! How fast is memset()!?!
https://stackoverflow.com/questions/3654905/faster-way-to-zero-memory-than-with-memset
Faster then finding the bug this will fix!s

What do we need to look for?

```
MATH/1.1 100 OK\nResult: 2\nRounding: False\nX-Server-
Version: 1.1.4\nOverflow: False\nConnection: Close\n\n
```

```
char *pStart = strstr(pBuffer, RESULT);

char *pEnd = strchr(pStart + 1, NEWLINE);

char *pDigit = strchr(pStart, SPACE) + 1;

*pEnd = '\0';

int value = atoi(pDigit); // error check!  strtol()?
// int value = my_atoi(pDigit, &bError);

*pEnd = '\n';
```

(Partial) Error check!

```
char *pStart = strstr(pBuffer, RESULT);
char *pEnd = strchr(pStart + 1, NEWLINE);
char *pDigit = strchr(pStart, SPACE) + 1;
*pEnd = '\0';

while( pDigit != pEnd )                 // error check
{
  if( !isdigit(*pDigit) )
  {
    //ERROR
  }
  pDigit ++;                            // move
}

pDigit = strchr(pStart, SPACE) + 1;  // reset
int value = atoi(pDigit);

*pEnd = '\n';

// don't forget -1 is legal!
```

What do we need to look for?

```
MATH/1.1 100 OK\nResult: 2\nRounding: False\nX-Server-
Version: 1.1.4\nOverflow: False\nConnection: Close\n\n
```

```
char szCloseOption[MAX];

char *pStart = strstr(pBuffer, CONNECTION);

char *pEnd = strchr(pStart + 1, NEWLINE);

char *pOption = strchr(pStart, SPACE) + 1;

*pEnd = '\0';

strncpy(&(szCloseOption[0]), pOption, MAX);

*pEnd = '\n';
```

What do we need to look for?

```
MATH/1.1 100 OK\nResult: 2\nRounding: False\nX-Server-Version: 1.1.4\nOverflow: False\nConnection: Close\n\n
```

```
char szXOption[MAX];

char *pStart = strstr(pBuffer,                    );

char *pEnd = strchr(pStart + 1, NEWLINE);

char *pOption = strchr(pStart, SPACE) + 1;

*pEnd = '\0';

strncpy(&(szXOption[0]), pOption, MAX);

*pEnd = '\n';
```