Command-Line Exercises Assignment 1

Basic Commands

1. strcount - Count characters in a string

Usage:

```
./strcount "hello"                # Output: 5
./strcount -w "hello world"       # Count words: 2
```

Options:

- `-h` : Show help
- `-c` : Count characters (default)
- `-w` : Count words

2. strupper - Convert string to uppercase

Usage:

```
./strupper "hello"                # Output: HELLO
./strupper -f input.txt           # Convert file contents
```

Options:

- `-h` : Show help
- `-f` : Read from file

3. strlower - Convert string to lowercase

Usage:

```
./strlower "HELLO"                # Output: hello
./strlower -f input.txt           # Convert file contents
```

Options:

- `-h` : Show help
- `-f` : Read from file

4. strrev - Reverse a string

Usage:

```
./strrev "hello"              # Output: olleh
./strrev -w "hello world"     # Reverse words: world hello
```

Options:

- `-h` : Show help
- `-w` : Reverse word order

---

5. strcat - Concatenate strings

Usage:

```
./strcat "hello" "world"        # Output: helloworld
./strcat -s " " "hello" "world" # Output: hello world
```

Options:

- `-h` : Show help
- `-s` : Separator character

---

6. strcmp - Compare two strings

Usage:

```
./strcmp "abc" "abc"            # Output: Equal
./strcmp "abc" "xyz"            # Output: Not equal
./strcmp -i "ABC" "abc"         # Case insensitive
```

Options:

- `-h` : Show help
- `-i` : Case insensitive comparison

---

7. strsub - Extract substring

Usage:

```
./strsub "hello" -s 1 -e 4      # Output: ell
./strsub "hello" -n 3           # First 3 chars: hel
./strsub "hello" -l 2           # Last 2 chars: lo
```

Options:

- `-h` : Show help
- `-s` : Start position
- `-e` : End position
- `-n` : First N characters
- `-l` : Last N characters

---

8. strfind - Find character in string

Usage:

```
./strfind "hello" -c 'l'          # Output: Found at position 2
./strfind "hello" -c 'x'          # Output: Not found
./strfind "hello" -c 'l' -a       # All positions: 2, 3
```

Options:

- `-h` : Show help
- `-c` : Character to find
- `-a` : Find all occurrences

Learning Goals: Linear search, strchr() function

---

9. strreplace - Replace character

Usage:

```
./strreplace "hello" -o 'l' -n 'x'  # Output: hexxo
./strreplace "hello" -o 'l' -n 'x' -f  # First only: hexlo
```

Options:

- `-h` : Show help
- `-o` : Old character
- `-n` : New character
- `-f` : Replace first occurrence only

---

10. strrepeat - Repeat string N times

Usage:

```
./strrepeat "abc" -n 3          # Output: abcabcabc
```

```
./strrepeat "Hi" -n 5 -s " "    # Output: Hi Hi Hi Hi Hi
```

Options:

- `-h` : Show help
- `-n` : Number of repetitions
- `-s` : Separator between repetitions

---

11. strvowel - Count vowels

Usage:

```
./strvowel "hello"              # Output: 2 vowels
./strvowel -i "HELLO"           # Case insensitive: 2 vowels
```

Options:

- `-h` : Show help
- `-i` : Case insensitive

---

12. strconsonant - Count consonants

Usage:

```
./strconsonant "hello"          # Output: 3 consonants
./strconsonant -i "HELLO"       # Case insensitive
```

Options:

- `-h` : Show help
- `-i` : Case insensitive

---

13. strpalindrome - Check if palindrome

Usage:

```
./strpalindrome "racecar"       # Output: Yes, palindrome
./strpalindrome "hello"         # Output: Not palindrome
./strpalindrome -i "RaceCar"    # Case insensitive
```

Options:

- `-h` : Show help
- `-i` : Case insensitive check

14. strspace - Count spaces

Usage:

```
./strspace "hello world"        # Output: 1 space
./strspace "a b c d"            # Output: 3 spaces
./strspace -a "hello\tworld"    # All whitespace: 1
```

Options:

- `-h` : Show help
- `-a` : Count all whitespace (tabs, newlines)

---

15. strdigit - Count digits in string

Usage:

```
./strdigit "hello123"           # Output: 3 digits
./strdigit "abc123xyz456"       # Output: 6 digits
```

Options:

- `-h` : Show help
- `-l` : List all digits found

Basic Arithmetic Operations (16-30)

16. calc - Simple calculator

Usage:

```
./calc -a 5 3                   # Add: 8
./calc -s 5 3                   # Subtract: 2
./calc -m 5 3                   # Multiply: 15
./calc -d 6 3                   # Divide: 2
./calc -p 2 3                   # Power: 8
```

Options:

- `-h` : Show help
- `-a` : Addition
- `-s` : Subtraction
- `-m` : Multiplication
- `-d` : Division

- `-p` : Power

---

17. sum - Sum of numbers

Usage:

```
./sum 1 2 3 4 5                # Output: 15
./sum -f numbers.txt           # Sum from file
```

Options:

- `-h` : Show help
- `-f` : Read numbers from file

---

18. avg - Average of numbers

Usage:

```
./avg 10 20 30                 # Output: 20.00
./avg -i 10 20 30              # Integer average: 20
```

Options:

- `-h` : Show help
- `-i` : Integer average
- `-f` : Floating point average (default)

---

19. max - Find maximum

Usage:

```
./max 5 10 3 8                 # Output: 10
./max -f numbers.txt           # Max from file
```

Options:

- `-h` : Show help
- `-f` : Read from file

---

20. min - Find minimum

Usage:

```
./min 5 10 3 8                # Output: 3
./min -f numbers.txt          # Min from file
```

Options:

- `-h` : Show help
- `-f` : Read from file

---

21. factorial - Calculate factorial

Usage:

```
./factorial 5                 # Output: 120
./factorial -r 5              # Recursive: 120
```

Options:

- `-h` : Show help
- `-r` : Use recursive algorithm
- `-i` : Use iterative algorithm (default)

---

22. power - Calculate power

Usage:

```
./power 2 3                   # Output: 8 (2^3)
./power -b 2 -e 10            # Output: 1024
```

Options:

- `-h` : Show help
- `-b` : Base number
- `-e` : Exponent

---

23. sqrt - Square root

Usage:

```
./sqrt 16                     # Output: 4
./sqrt -p 2 25                # Precision 2: 5.00
```

Options:

- `-h` : Show help
- `-p` : Decimal precision

---

24. even - Check if even

Usage:

```
./even 4                          # Output: Yes, even
./even 5                          # Output: No, odd
./even -r 1 10                    # Range: 2,4,6,8,10
```

Options:

- `-h` : Show help
- `-r` : Check range

---

25. prime - Check if prime

Usage:

```
./prime 7                         # Output: Yes, prime
./prime 8                         # Output: Not prime
./prime -r 1 20                   # List primes: 2,3,5,7,11,13,17,19
```

Options:

- `-h` : Show help
- `-r` : Find primes in range

---

26. fibo - Fibonacci sequence

Usage:

```
./fibo 7                          # Output: 0 1 1 2 3 5 8
./fibo -n 10                      # First 10: 0 1 1 2 3 5 8 13 21 34
```

Options:

- `-h` : Show help
- `-n` : Number of terms

---

27. digits - Count digits in number

Usage:

```
./digits 12345                  # Output: 5 digits
./digits -s 12345               # Sum: 15
```

Options:

- `-h` : Show help
- `-s` : Sum of digits

---

28. reverse - Reverse a number

Usage:

```
./reverse 12345                 # Output: 54321
./reverse -c 12321              # Check palindrome: Yes
```

Options:

- `-h` : Show help
- `-c` : Check if palindrome

---

29. strsum - Sum of digits in string

Usage:

```
./strsum "abc123xyz456"         # Output: 21 (1+2+3+4+5+6)
./strsum "hello123"             # Output: 6
```

Options:

- `-h` : Show help
- `-v` : Verbose (show which digits found)

---

30. stralpha - Count alphabets

Usage:

```
./stralpha "hello123"           # Output: 5 alphabets
./stralpha -u "Hello"           # Uppercase only: 1
./stralpha -l "Hello"           # Lowercase only: 4
```

Options:

- `-h` : Show help
- `-u` : Count uppercase only
- `-l` : Count lowercase only

---

31. stralphadigit - Count alpha and digits separately

Usage:

```
./stralphadigit "hello123"      # Output: Alpha: 5, Digits: 3
./stralphadigit "test456!@#"    # Alpha: 4, Digits: 3, Other: 3
```

Options:

- `-h` : Show help
- `-a` : Show all character types

---

32. strascii - Show ASCII values

Usage:

```
./strascii "ABC"                # Output: A=65, B=66, C=67
./strascii -d 65                # Decode: A
```

Options:

- `-h` : Show help
- `-d` : Decode ASCII value to character

---

33. strhex - Convert string to hex

Usage:

```
./strhex "ABC"                  # Output: 41 42 43
./strhex -d "41 42 43"          # Decode: ABC
```

Options:

- `-h` : Show help
- `-d` : Decode hex to string

Learning Goals: Hexadecimal conversion, printf formatting

---

34. numtostr - Number to words

Usage:

```
./numtostr 123                # Output: one two three
./numtostr -w 123             # Words: one hundred twenty three
```

Options:

- `-h` : Show help
- `-w` : Full words (not digit by digit)

Learning Goals: Number to string conversion, arrays

---

35. strtonum - Extract numbers from string

Usage:

```
./strtonum "abc123xyz456"     # Output: 123, 456
./strtonum -s "price: $99.99" # Output: 99.99
```

Options:

- `-h` : Show help
- `-s` : Sum extracted numbers

Learning Goals: Pattern extraction, number parsing

---

36. caesar - Caesar cipher encryption

Usage:

```
./caesar "hello" -k 3         # Output: khoor (shift by 3)
./caesar "khoor" -k -3        # Decrypt: hello
```

Options:

- `-h` : Show help
- `-k` : Shift key (positive or negative)

Learning Goals: Cipher algorithms, modular arithmetic

---

37. strfreq - Character frequency

Usage:

```
./strfreq "hello"                # Output: h:1, e:1, l:2, o:1
./strfreq -s "hello"             # Sorted: e:1, h:1, l:2, o:1
```

Options:

- `-h` : Show help
- `-s` : Sort by character
- `-f` : Sort by frequency

Learning Goals: Frequency counting, arrays/maps

---

38. strunique - Count unique characters

Usage:

```
./strunique "hello"              # Output: 4 unique chars
./strunique -l "hello"           # List: h, e, l, o
```

Options:

- `-h` : Show help
- `-l` : List unique characters

Learning Goals: Unique element detection, sets

---

39. strduplicate - Find duplicate characters

Usage:

```
./strduplicate "hello"           # Output: l appears 2 times
./strduplicate -a "hello world" # All duplicates
```

Options:

- `-h` : Show help
- `-a` : Show all duplicates

Learning Goals: Duplicate detection, frequency analysis

---

40. strfirst - First N characters

Usage:

```
./strfirst "hello" -n 3          # Output: hel
./strfirst "hello world" -w 2    # First 2 words: hello world
```

Options:

- `-h` : Show help
- `-n` : Number of characters
- `-w` : Number of words

Learning Goals: String slicing, boundary checking

---

41. strlast - Last N characters

Usage:

```
./strlast "hello" -n 3          # Output: llo
./strlast "hello world" -w 1    # Last word: world
```

Options:

- `-h` : Show help
- `-n` : Number of characters
- `-w` : Number of words

---

42. wordcount - Count words in string

Usage:

```
./wordcount "hello world from C"    # Output: 4 words
./wordcount -l "hello\nworld"       # Lines: 2
```

Options:

- `-h` : Show help
- `-w` : Count words (default)
- `-l` : Count lines
- `-c` : Count characters

---

43. strremove - Remove character from string

Usage:

```
./strremove "hello" -c 'l'      # Output: heo
./strremove "hello" -c 'l' -f   # First only: helo
```

Options:

- `-h` : Show help
- `-c` : Character to remove
- `-f` : Remove first occurrence only

---

44. triangle - Print number triangle

Usage:

```
./triangle 5
# Output:
# 1
# 1 2
# 1 2 3
# 1 2 3 4
# 1 2 3 4 5

./triangle -r 5                 # Right aligned
./triangle -i 5                 # Inverted
```

Options:

- `-h` : Show help
- `-r` : Right aligned
- `-i` : Inverted triangle

---

45. table - Multiplication table

Usage:

```
./table 5 -n 10
# Output: 5x1=5, 5x2=10, ... 5x10=50

./table -r 1 10                 # Full table 1-10
```

Options:

- `-h` : Show help
- `-n` : Number of rows
- `-r` : Range (from-to)

---

46. pattern - Print patterns

Usage:

```
./pattern -s '*' -n 5
# Output:
# *
# **
# ***
# ****
# *****

./pattern -s '#' -n 4 -p pyramid
# Output:
#    #
#   ###
#  #####
# #######
```

Options:

- `-h` : Show help
- `-s` : Symbol to use
- `-n` : Number of rows
- `-p` : Pattern type (triangle, pyramid, diamond)

---

47. armstrong - Check Armstrong number

Usage:

```
./armstrong 153                # Output: Yes (1^3 + 5^3 + 3^3 = 153)
./armstrong 123                # Output: No
./armstrong -r 1 1000          # Find all in range
```

Options:

- `-h` : Show help
- `-r` : Find in range
- `-v` : Verbose (show calculation)

---

48. perfect - Check perfect number

Usage:

```
./perfect 28                   # Output: Yes (1+2+4+7+14=28)
./perfect 12                   # Output: No
./perfect -r 1 1000            # Find all perfect numbers
```

Options:

- `-h` : Show help
- `-r` : Find in range
- `-v` : Verbose (show divisors)

## Example Template to Start

```c
#include <stdio.h>
#include <stdlib.h>
#include <getopt.h>

void print_usage(const char *prog) {
    printf("Usage: %s [OPTIONS]\n", prog);
    printf("Options:\n");
    printf("  -h, --help     Show help\n");
    // Add more options
}

int main(int argc, char *argv[]) {
    int opt;

    static struct option long_options[] = {
        {"help", no_argument, 0, 'h'},
        {0, 0, 0, 0}
    };

    while ((opt = getopt_long(argc, argv, "h", long_options, NULL)) != -1) {
        switch (opt) {
            case 'h':
                print_usage(argv[0]);
                exit(0);
            default:
                print_usage(argv[0]);
                exit(1);
        }
    }

    // Your program logic

    return 0;
}
```