COMP1511 18s2 (webcms)

Code Examples from Lectures on reading_and_writing_files

create_file.c

Simple example of file creation creates file "andrew.txt" containing 1 line ("Andrew rules!")

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

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

    outputStream = fopen("andrew.txt", "a");
    if (outputStream == NULL) {
        fprintf(stderr, "%s: open of '%s' failed\n", argv[0], "andrew.txt");
        return 1;
    }

    fprintf(outputStream, "Andrew rules!\n");
    fclose(outputStream);

    return 0;
}
```

append_file.c

Simple example of file creation appends to file "andrew.txt" the line "Andrew rules!"

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

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

    outputStream = fopen("andrew.txt", "a");
    if (outputStream == NULL) {
        perror("andrew.txt"); // prints why the open failed
        return 1;
    }

    fprintf(outputStream, "Andrew rules!\n");
    fclose(outputStream);

    return 0;
}
```

cat_fgetc.c

Simple implementation of reading files passed as command line arguments using fgetc - in other words cat

```c
#include <stdio.h>

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

    // go through a list of files specified on the command line
    // printing their contents in turn to standard output

    int argument = 1;
    while (argument < argc) {

        // FILE is an opaque (hidden type) defined in stdio.h
        // "r" indicate we are opening file to read contents

        FILE *stream = fopen(argv[argument], "r");
        if (stream == NULL) {
            perror(argv[argument]);  // prints why the open failed
            return 1;
        }

        // fgetc returns next the next byte (ascii code if its a text file)
        // from a stream
        // it returns the special value EOF if not more bytes can be read from the stream

        int c = fgetc(stream);

        // return  bytes from the stream (file) one at a time

        while (c != EOF) {
            fputc(c, stdout); // write the byte to standard output
            c = fgetc(stream);
        }

        argument = argument + 1;
    }
    return 0;
}
```

cat_fgets.c

Simple implementation of reading files passed as command line arguments using fgets - in other words cat

```c
#include <stdio.h>
#define MAX_LINE 1024

int main(int argc, char *argv[]) {
    int argument = 1;
    while (argument < argc) {

        FILE *stream = fopen(argv[argument], "r");
        if (stream == NULL) {
            perror(argv[argument]);  // prints why the open failed
            return 1;
        }

        char line[MAX_LINE];
        while (fgets(line, MAX_LINE, stream) != NULL) {
            printf("%s", line);
        }

        argument = argument + 1;
    }
    return 0;
}
```

copy.c

Simple example of cp command copy contents of file specified as first argument to file specified as 2nd argument 8/5//18

```c
#include <stdio.h>

int main(int argc, char *argv[]) {
    if (argc != 2) {
        fprintf(stderr, "Usage: %s <source file> <destination file>\n", argv[0]);
        return 1;
    }

    FILE *inputStream = fopen(argv[1], "r");
    if (inputStream == NULL) {
        perror(argv[1]);  // prints why the open failed
        return 1;
    }

    FILE *outputStream = fopen(argv[2], "w");
    if (outputStream == NULL) {
        // perror could be used for a better error message here
        fprintf(stderr, "%s: open of '%s' failed\n", argv[0], argv[2]);
        return 1;
    }

    int c = fgetc(inputStream);
    while (c != EOF) {
        fputc(c, outputStream);
        c = fgetc(inputStream);
    }
    fclose(outputStream);

    return 0;
}
```

wc_fgetc0.c

Simple implementation of Unix wc command
It counts lines, word & characters in the file specified as an argument

This version uses fgetc, see also wc_fgets.c
See later version for more sophisticated argument handling

```c
#include <stdio.h>
#include <ctype.h>

#define MAX_LINE 4096

void process_stream(FILE *stream, char stream_name[]);

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

    if (argc != 2) {
        fprintf(stderr, "Usage: %s <filename>\n", argv[0]);
        return 1;
    }

    FILE *in = fopen(argv[1], "r");
    if (in == NULL) {
        // perror could be used for a better error message here
        fprintf(stderr, "%s: open of '%s' failed\n", argv[0], argv[1]);
        return 1;
    }

    process_stream(in, argv[1]);

    return 0;
}

void process_stream(FILE *stream, char stream_name[]) {
    int line_count = 0;
    int character_count = 0;
    int word_count = 0;

    int lastc = ' ';
    int c = fgetc(stream);

    while (c != EOF) {
        if (c == '\n') {
            line_count = line_count + 1;
        }
        if (isspace(c) && !isspace(lastc)) {
            word_count = word_count + 1;
        }
        character_count = character_count + 1;
```

wc_fgetc1.c

/ Simple implementation of Unix wc command
It counts lines, word & characters from the files specified as arguments

If no files are specified instead stdin is processed
This allows use in a pipeline, e.g.: % dcc grep.c -o grep % dcc wc_fgetc1.c -o wc % ./grep return grep.c grep.c:30: return 1; grep.c:38: return 1; grep.c:44: return 0; % ./grep return grep.c | ./wc <stdin> contains 3 lines 9 words 85 characters

```c
#include <stdio.h>
#include <ctype.h>

#define MAX_LINE 4096

void process_stream(FILE *stream, char stream_name[]);

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

    if (argc == 1) {
        // if no files are specified, process stdin
        process_stream(stdin, "<stdin>");
    } else {
        int argument = 1;
        while (argument < argc) {
            FILE *in = fopen(argv[argument], "r");
            if (in == NULL) {
                // perror could be used for a better error message here
                fprintf(stderr, "%s: open of '%s' failed\n", argv[0], argv[argument]);
                return 1;
            }

            process_stream(in, argv[argument]);

            argument = argument + 1;
        }
    }

    return 0;
}

void process_stream(FILE *stream, char stream_name[]) {
    int line_count = 0;
    int character_count = 0;
    int word_count = 0;

    int lastc = ' ';
    int c = fgetc(stream);

    while (c != EOF) {
        if (c == '\n') {
            line_count = line_count + 1;
```

wc_fgets.c

Simple implementation of Unix wc command
It counts lines, word & characters in the file specified as an argument
This version uses fgets

```c
#include <stdio.h>
#include <ctype.h>

#define MAX_LINE 4096

void process_stream(FILE *stream, char stream_name[]);

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

    if (argc != 2) {
        fprintf(stderr, "Usage: %s <filename>\n", argv[0]);
        return 1;
    }

    in = fopen(argv[1], "r");
    if (in == NULL) {
        // perror could be used for a better error message here
        fprintf(stderr, "%s: open of '%s' failed\n", argv[0], argv[1]);
        return 1;
    }
    process_stream(in, argv[1]);

    return 0;
}

void process_stream(FILE *stream, char stream_name[]) {
    char line[MAX_LINE];

    int line_count = 0;
    int character_count = 0;
    int word_count = 0;

    while (fgets(line, MAX_LINE, stream) != NULL) {
        int i = 0;
        while (line[i] != '\0') {
            if (i > 0 && isspace(line[i]) && !isspace(line[i - 1])) {
                word_count = word_count + 1;
            }
            character_count = character_count + 1;
            i = i + 1;
        }
```

grep.c

Print lines containing specified pattern from the files specified as arguments
This version provides the unix filter behaviour where if no files are specified it process stdin

Note this code will produce incorrect results if matched lines contain >= MAX_LINE characters. Fixing this left as an exercise

```c
#include <stdio.h>
#include <string.h>

#define MAX_LINE 65536

void search_stream(FILE *stream, char stream_name[], char search_for[]);

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

    if (argc < 2) {
        fprintf(stderr, "Usage: %s <prefix> <files>\n", argv[0]);
        return 1;
    } if (argc == 2) {
        search_stream(stdin, "<stdin>", argv[1]);
    } else {

        int argument = 2;
        while (argument < argc) {
            FILE *in = fopen(argv[argument], "r");
            if (in == NULL) {
                // perror could be used for a better error message here
                fprintf(stderr, "%s: open of '%s' failed\n", argv[0], argv[argument]);
                return 1;
            }

            search_stream(in, argv[argument], argv[1]);
            argument = argument + 1;
        }

    }

    return 0;
}

void search_stream(FILE *stream, char stream_name[], char search_for[]) {
    char line[MAX_LINE];

    int line_number = 1;
    while (fgets(line, MAX_LINE, stream) != NULL) {
        if (strstr(line, search_for) != NULL) {
            printf("%s:%d:%s", stream_name, line_number, line);
        }
```

prefix.c

Print lines starting with specified string from specified files

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

#define MAX_LINE 1024

int is_prefix(char prefix[], char str[]);

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

    if (argc < 2) {
        fprintf(stderr, "Usage: %s <prefix> <files>\n", argv[0]);
        exit(1);
    }

    char *searchString = argv[1];
    int argument = 2;
    while (argument < argc) {

        FILE *stream = fopen(argv[argument], "r");
        if (stream == NULL) {
            // perror could be used for a better error message here
            fprintf(stderr, "%s: open of '%s' failed\n", argv[0], argv[argument]);
            return 1;
        }

        char line[MAX_LINE];
        while (fgets(line, MAX_LINE, stream) != NULL) {
            if (is_prefix(searchString, line)) {
                printf("%s", line);
            }
        }

        fclose(stream);
        argument = argument + 1;
    }
    return 0;
}

// test if str starts with prefix
int is_prefix(char prefix[], char str[]) {
    int length = strlen(prefix);
```

replace.c

print lines form file after replacing specified pattern after replacing specified pattern

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

#define MAX_LINE 1024
#define MAX_REPLACEMENT_LINE 32768

void replace(char string[], char target[], char replacement[], char new_string[], int new_string_len);

int main(int argc, char *argv[]) {
    char line[MAX_LINE];
    char changed_line[MAX_REPLACEMENT_LINE];

    if (argc < 3) {
        fprintf(stderr, "Usage: %s <target> <replacement> <files>\n", argv[0]);
        exit(1);
    }

    char *target_string = argv[1];
    char *replacement_string = argv[2];

    int argument = 0;
    while (argument < argc) {
        FILE *stream = fopen(argv[argument], "r");
        if (stream == NULL) {
            // perror could be used for a better error message here
            fprintf(stderr, "%s: open of '%s' failed\n", argv[0], argv[argument]);
            return 1;
        }

        while (fgets(line, MAX_LINE, stream) != NULL) {
            replace(line, target_string, replacement_string, changed_line, MAX_REPLACEMENT_LINE);
            printf("%s", changed_line);
        }

        argument = argument + 1;
    }

    return 0;
}

// copy string to new_string replacing all instances of target with replacement
```