

# Software Construction

## [cat.c](#)

Simple /bin/cat emulation.

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

// write bytes of stream to stdout
void process_stream(FILE *in){
    while (1){
        int ch = fgetc(in);
        if (ch == EOF){
            break;
        }
        if (fputc(ch, stdout) == EOF){
            fprintf(stderr, "cat:");
            perror("");
            exit(1);
        }
    }
}

// process files given as arguments
// if no arguments process stdin
int main(int argc, char *argv[]){
    if (argc == 1){
        process_stream(stdin);
    }
    else
        for (int i = 1; i < argc; i++){
            FILE *in = fopen(argv[i], "r");
            if (in == NULL){
                fprintf(stderr, "%s: %s: ", argv[0], argv[i]);
                perror("");
                return 1;
            }
            process_stream(in);
            fclose(in);
        }
    return 0;
}
```

## [WC.C](#)

Simple /usr/bin/wc emulation.

```

#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>

// count lines, words, chars in stream
void process_stream(FILE *in){
    int n_lines = 0, n_words = 0, n_chars = 0;
    int in_word = 0, c;
    while ((c = fgetc(in)) != EOF){
        n_chars++;
        if (c == '\n'){
            n_lines++;
            if (isspace(c)){
                in_word = 0;
            } else if (!in_word){
                in_word = 1;
                n_words++;
            }
        }
    }
    printf("%6d %6d %6d", n_lines, n_words, n_chars);
}

// process files given as arguments
// if no arguments process stdin
int main(int argc, char *argv[]){
    if (argc == 1){
        process_stream(stdin);
    } else
        for (int i = 1; i < argc; i++){
            FILE *in = fopen(argv[i], "r");
            if (in == NULL){
                fprintf(stderr, "%s: %s: ", argv[0], argv[i]);
                perror("");
                return 1;
            }
            process_stream(in);
            printf(" %s\n", argv[i]);
            fclose(in);
        }
    return 0;
}

```

[grep.c](#)

Over-simple /usr/bin/grep emulation.

```

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

// print lines containing the specified substring
// breaks on long lines, does not implement regexs or other grep features
void process_stream(FILE *stream, char *stream_name, char *substring){
    char line[65536];
    int line_number = 1;
    while (fgets(line, sizeof line, stream) != NULL){
        if (strstr(line, substring) != NULL)
            printf("%s:%d:%s", stream_name, line_number, line);
        line_number = line_number + 1;
    }
}

// process files given as arguments
// if no arguments process stdin
int main(int argc, char *argv[]){
    if (argc == 2)
        process_stream(stdin, "<stdin>", argv[1]);
    else
        for (int i = 2; i < argc; i++){
            FILE *in = fopen(argv[i], "r");
            if (in == NULL){
                fprintf(stderr, "%s: %s: ", argv[0], argv[i]);
                perror("");
                return 1;
            }
            process_stream(in, argv[i], argv[1]);
            fclose(in);
        }
    return 0;
}

```

[uniq.c](#)

Over-simple /usr/bin/uniq emulation.

```

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

#define MAX_LINE 65536

// cope stream to stdout except for repeated lines
void process_stream(FILE *stream){
    char line[MAX_LINE];
    char lastLine[MAX_LINE];
    int line_number = 0;

    while (fgets(line, MAX_LINE, stdin) != NULL){
        if (line_number == 0 || strcmp(line, lastLine) != 0){
            fputs(line, stdout);
            strncpy(lastLine, line, MAX_LINE);
        }
        line_number++;
    }
}

// process files given as arguments
// if no arguments process stdin
int main(int argc, char *argv[]){
    if (argc == 1)
        process_stream(stdin);
    else
        for (int i = 1; i < argc; i++){
            FILE *in = fopen(argv[i], "r");
            if (in == NULL){
                fprintf(stderr, "%s: %s: ", argv[0], argv[i]);
                perror("");
                return 1;
            }
            process_stream(in);
            fclose(in);
        }
    return 0;
}

```

**COMP(2041|9044) 20T2: Software Construction** is brought to you by  
the [School of Computer Science and Engineering](#)  
at the [University of New South Wales](#), Sydney.

For all enquiries, please email the class account at [cs2041@cse.unsw.edu.au](mailto:cs2041@cse.unsw.edu.au)

CRICOS Provider 00098G