# 1. Makefile

**Step I:** Write a C program that sums two numbers given as the input parameters and prints the sum on the screen (or does some other small operation of your choice). Divide the program into three parts: the main part, the function part and the header file. Write a Makefile to compile the files with so that each .c file is compiled separately and then linked together at the end. Running 'make' on the command line compiles the program. Also remember to add file dependencies to the Makefile.

**Step II:** Add variable int c into the structure in header file and run make to test what files are compiled. Then change the program so that instead of the sum the program calculates some further value, for example multiplication. Then run 'make' and note which files are compiled.

**Vastaus**

Main.c

```c
#include "function.h"
#include <stdio.h>

int main () {
    float num1, num2, rslt;
    printf("Give first number: ");
    scanf("%f", &num1);
    printf("Give second number: ");
    scanf("%f", &num2);
    rslt = summa(num1, num2);
    printf("The sum of %.2f and %.2f is %.2f \r\n", num1, num2, rslt);
    return 0;
}
```

function.c

```c
#include "function.h"

float summa (float num1, float num2) {
    float sum;
    sum = num1 * num2;
    return sum;
}
```

function.h

```c
float summa (float num1, float num2);
int c;
```

Makefile

```
MyProgram: Main.o function.o
    gcc -o MyProgram Main.o function.o

main.o: Main.c function.h
    gcc -c Main.c

function.o: function.c function.h
    gcc -c function.c
```

make:n suorittaminen kokosi uudestaan, kun muutettiin header tiedostoa sekä, kun muutettiin functio tiedoston plus lasku kertolaskuksi.

# 2. Command line parameters

Write a C program that prints all its command line parameters and also the environment variables.

**Vastaus**

```c
// how to list env
// https://stackoverflow.com/questions/2085302/printing-all-environment-variables-in-c-c

#include <stdio.h>

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

    if (argc < 2 ) {
        printf("Expected at least one argument!\r\n");
    } else {
        int i = 1;
        while (i < argc) {
            printf("Argument %d: %s\r\n", i, argv[i]);
            i++;
        }
        for (char **env = envp; *env != 0; env++) {
            char *thisEnv = *env;
            printf("%s\n", thisEnv);
         }

    }
    return 0;
}
```

Output

```
student@ubuntu:~/Desktop/Kurssimateriaali/Viikko_9/task2$ ./2
Expected at least one argument!
student@ubuntu:~/Desktop/Kurssimateriaali/Viikko_9/task2$ ./2 Tässä muutamia argumenttejä
Argument 1: Tässä
Argument 2: muutamia
Argument 3: argumenttejä
SHELL=/bin/bash
SESSION_MANAGER=local/ubuntu:@/tmp/.ICE-unix/1624,unix/ubuntu:/tmp/.ICE-unix/1624
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
XDG_MENU_PREFIX=gnome-
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
LC_ADDRESS=fi_FI.UTF-8
GNOME_SHELL_SESSION_MODE=ubuntu
LC_NAME=fi_FI.UTF-8
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
```

# 3. Environment

Write a program that after receiving some operands as command line parameters, fetches the operator from some particular environment variable, and calculates the result. For example:

```
> export OPERATOR=add
> ./mycalc 1 2 3
6
```

You may find the library function `getenv` useful. Implement your program so that it can handle the addition and multiplication operator with any number of operands.

**Vastaus**

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

int main(int argc, char **argv) {
    int i;
    char * op;
    double result;

    if ( (op = getenv("OPERATOR")) == NULL) {
        fprintf(stderr, "environment variable OPERATOR not defined\n");
        exit(1);
    }
    printf("Operator: %s\n", op);
    if (argc>1) {
        result=atof(argv[1]);
        for (i=2; i<argc; i++) {
            if (strcmp(op,"add")==0) {
                result += atof(argv[i]);
            }
            if (strcmp(op,"multiply")==0) {
                result *= atof(argv[i]);
            }
            if (strcmp(op,"subtract")==0) {
                result -= atof(argv[i]);
            }
            if (strcmp(op,"divide")==0) {
                result /= atof(argv[i]);
            }
        }
    } else {
        result=0;
    }

    printf ("result with OPERATOR %s is %f\n", op, result);

    return 0;
}
```

Output

```
student@ubuntu:~/Desktop/Kurssimateriaali/Viikko_9/task3$ export OPERATOR=add
student@ubuntu:~/Desktop/Kurssimateriaali/Viikko_9/task3$ ./3a 3 2 5
Operator: add
result with OPERATOR add is 10.000000
student@ubuntu:~/Desktop/Kurssimateriaali/Viikko_9/task3$ export OPERATOR=multiply
student@ubuntu:~/Desktop/Kurssimateriaali/Viikko_9/task3$ ./3a 3 2 5
Operator: multiply
result with OPERATOR multiply is 30.000000
student@ubuntu:~/Desktop/Kurssimateriaali/Viikko_9/task3$ export OPERATOR=subtract
student@ubuntu:~/Desktop/Kurssimateriaali/Viikko_9/task3$ ./3a 3 2 5
Operator: subtract
result with OPERATOR subtract is -4.000000
```

# 4. Reading input

Some filters present the problem that you cannot process the input stream in reverse direction. Write a program that reads lines from the standard input (stdin) into a doubly linked list, and prints those lines to the standard output (stdout) in reverse order. Make the program modular using subroutines. You can start with the given example code [task4s.c] and just insert the important missing lines.

## Vastaus

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

#define MAXLEN 1024

struct lines {
    char * string;
    struct lines * prev;
    struct lines * next;
};

int read_lines(struct lines ** line, FILE * stream) {
    char tmp[MAXLEN];
    struct lines * lp;
    struct lines * temp;

    while (fgets(tmp, MAXLEN, stream) != NULL) {

        if (*line == NULL) {
            if ((*line = lp = malloc(sizeof(struct lines))) == NULL) {
                fprintf(stderr, "ERROR: could not allocate memory\n");
                return -1;
            }
            lp->prev = lp->next = NULL;
        } else {
            if ( (lp->next = malloc(sizeof(struct lines))) == NULL) {
                fprintf(stderr, "ERROR: could not allocate memory\n");
                return -1;
            }

            /* Here you should insert the lines that link the list correctly */

            lp->next->next = NULL;
            temp = lp;
            lp = lp->next;
            lp->prev = temp;
            *line = lp;
        }
        if ( (lp->string = malloc(strlen(tmp) + 1)) == NULL) {
                fprintf(stderr, "ERROR: could not allocate memory\n");
                return -1;
        }
        strcpy(lp->string, tmp);
    }

    return 0;
}
```

```c
void print_lines(struct lines * line, FILE * stream) {
    struct lines * lp = line;

    /* Here you should implement a loop that prints the lines in reverse order.     */
    /* First you need to follow the links from the first node to the last node. */

    while(lp->next != NULL) {
        lp = lp->next;
    }
    printf("%s", lp->string);
    while(lp->prev != NULL) {
        lp = lp->prev;
        printf("%s", lp->string);
    }
}

void delete_lines(struct lines * line) {
    struct lines * lp;

    lp = line;
    while (lp != NULL) {
        line = lp->next;
        free(lp->string);
        free(lp);
        lp = line;
    }
}

int main(void) {
    struct lines * line = NULL;

    if (read_lines(&line, stdin) == -1)
        exit(1);
    print_lines(line, stdout);
    delete_lines(line);

    return 0;
}
```

Output

```
student@ubuntu:~/Desktop/Kurssimateriaali/Viikko_9/task4$ ./task4
Tässä
vähän
input
streamia
streamia
input
vähän
Tässä
```

# 5. Reading directory

Write a simple equivalent to '`ls`' that reads the contents of a directory file. The index node contains the most important information like the number of the links, the file size, the access permissions, and the last date of a modification. If you give the program an option `-r`, it will print a recursive listing of all subdirectories. You can start with the given code [task5s.c] and insert the important missing lines. And as the first version, skip recursion and list only the contents of the given directory.

**Vastaus**

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <dirent.h>
#include <sys/stat.h>
#include <time.h>
#include <errno.h>

#define MAXPATH 256

int list_dir(char * dir, int rec) {
    DIR * dirp;
    char path[MAXPATH];
    struct dirent *direntp;
    struct stat st;

    /* Here you should open the directory indicated by the given path */
    /* check functions opendir */
    if ((dirp = opendir(dir)) == NULL) {
        perror("opendir() error");
    } else {
        printf("Content of the current folder:\r\n");
        while ((direntp = readdir(dirp)) != NULL) {
            stat(direntp->d_name, &st);
            if (S_ISDIR(st.st_mode) != 0 && (strcmp(direntp->d_name, "..")) != 0 && (strcmp(direntp->d_name, ".")) != 0 && rec == 1) {
                printf("Opening directory: %s \r\n", direntp->d_name);
                strcpy(path, dir);
                strcat(path, "/");
                strcat(path, direntp->d_name);
                list_dir(path, rec);
            }
            printf("Name: %s    Links: %ld  Size: %ld   AccessPermissions: %d   LastModified: %ld \r\n", direntp->d_name, st.st_nlink, st.st_size, st.st_mode, st.st_mtime);
        }
    } if (closedir(dirp) == -1) {
        perror(dir);
        return -1;
    }
    return 0;
}

int main(int argc, char * * argv) {
    int i = 1, rec = 0;

    if (argc > 1) {
        if (strcmp(argv[i], "-r") == 0) {
            rec = 1;
            i++;
        }
    }
    while (i < argc) {
        if (list_dir(argv[i], rec) == -1) exit(1);
        i++;
    }

    return 0;
}
```

Output

Jostain syystä vaikka koodissa tarkistetaan onko kyseessä kansio vai ei haluaa jostain syystä avata .txt ja .pdf tiedoston kansiona, joka tietysti epäonnistuu. Muuten toimii halutulla tavalla ja assarin koneella samaa virhettä ei ollut.

```
student@ubuntu:~/Desktop/Kurssimateriaali/Viikko_9/task5$ ./task5 .
Content of the current folder:
Name: ..        Links: 9        Size: 4096      AccessPermissions: 16893       LastModified: 1647888779
Name: . Links: 3        Size: 4096      AccessPermissions: 16893        LastModified: 1647891435
Name: task5     Links: 1        Size: 17280     AccessPermissions: 33277       LastModified: 1647891435
Name: folder    Links: 3        Size: 4096      AccessPermissions: 16893       LastModified: 1647890095
Name: task5s.c  Links: 1        Size: 1350      AccessPermissions: 33152       LastModified: 1647891429
```

-r

```
student@ubuntu:~/Desktop/Kurssimateriaali/Viikko_9/task5$ ./task5 -r .
Content of the current folder:
Name: ..        Links: 9        Size: 4096      AccessPermissions: 16893        LastModified: 1647888779
Name: . Links: 3        Size: 4096      AccessPermissions: 16893        LastModified: 1647891435
Name: task5     Links: 1        Size: 17280     AccessPermissions: 33277        LastModified: 1647891435
Opening directory: folder
Content of the current folder:
Name: ..        Links: 9        Size: 4096      AccessPermissions: 16893        LastModified: 1647888779
Opening directory: 2ndFolder
Content of the current folder:
Name: ..        Links: 9        Size: 4096      AccessPermissions: 16893        LastModified: 1647888779
Name: . Links: 3        Size: 4096      AccessPermissions: 16893        LastModified: 1647891435
Opening directory: luento_teoriaosa.pdf
opendir() error: Not a directory
./folder/2ndFolder/luento_teoriaosa.pdf: Invalid argument
Name: luento_teoriaosa.pdf      Links: 3        Size: 4096      AccessPermissions: 16893        LastModified: 1647891435
Name: 2ndFolder Links: 9        Size: 4096      AccessPermissions: 16893        LastModified: 1647888779
Name: . Links: 3        Size: 4096      AccessPermissions: 16893        LastModified: 1647891435
Opening directory: Untitled Document 1
opendir() error: Not a directory
./folder/Untitled Document 1: Invalid argument
Name: Untitled Document 1       Links: 3        Size: 4096      AccessPermissions: 16893        LastModified: 1647891435
Name: folder    Links: 3        Size: 4096      AccessPermissions: 16893        LastModified: 1647890095
Name: task5s.c  Links: 1        Size: 1350      AccessPermissions: 33152        LastModified: 1647891429
```