Задача 1  
 Використайте popen(), щоб передати вивід команди rwho (команда UNIX) до more (команда UNIX) у програмі на C.

#include <stdio.h>

int main() {

    FILE \*fp;

    char buffer[128];

    fp = popen("rwho | more", "r");

    if (fp == NULL) {

        perror("error executing command");

        return 1;

    }

    while (fgets(buffer, sizeof(buffer), fp) != NULL) {

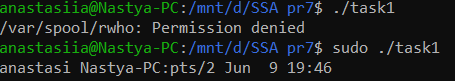
        printf("%s", buffer);

    }

    pclose(fp);

    return 0;

}



Задача 2  
 Напишіть програму мовою C, яка імітує команду ls -l в UNIX — виводить список усіх файлів у поточному каталозі та перелічує права доступу тощо.  
 (Варіант вирішення, що просто виконує ls -l із вашої програми, — не підходить.)

#include <stdio.h>

#include <stdlib.h>

#include <dirent.h>

#include <sys/stat.h>

#include <unistd.h>

#include <pwd.h>

#include <grp.h>

#include <time.h>

void print\_permissions(mode\_t mode) {

    printf("%c%c%c%c%c%c%c%c%c ",

           (S\_ISDIR(mode)) ? 'd' : '-',

           (mode & S\_IRUSR) ? 'r' : '-',

           (mode & S\_IWUSR) ? 'w' : '-',

           (mode & S\_IXUSR) ? 'x' : '-',

           (mode & S\_IRGRP) ? 'r' : '-',

           (mode & S\_IWGRP) ? 'w' : '-',

           (mode & S\_IXGRP) ? 'x' : '-',

           (mode & S\_IROTH) ? 'r' : '-',

           (mode & S\_IWOTH) ? 'w' : '-');

}

void list\_files(const char \*path) {

    struct dirent \*entry;

    struct stat file\_stat;

    DIR \*dir = opendir(path);

    if (!dir) {

        perror("Error opening directory");

        return;

    }

    while ((entry = readdir(dir)) != NULL) {

        char full\_path[1024];

        snprintf(full\_path, sizeof(full\_path), "%s/%s", path, entry->d\_name);

        if (stat(full\_path, &file\_stat) == 0) {

            print\_permissions(file\_stat.st\_mode);

            printf("%c ", (file\_stat.st\_mode & S\_IXOTH) ? 'x' : '-');

            printf("%ld ", file\_stat.st\_nlink);

            printf("%s ", getpwuid(file\_stat.st\_uid)->pw\_name);

            printf("%s ", getgrgid(file\_stat.st\_gid)->gr\_name);

            printf("%ld ", file\_stat.st\_size);

            printf("%.12s ", ctime(&file\_stat.st\_mtime) + 4);

            printf("%s\n", entry->d\_name);

        }

    }

    closedir(dir);

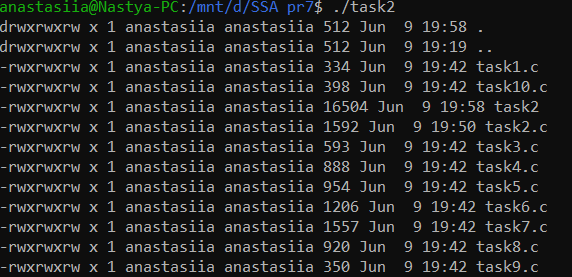
}

int main() {

    list\_files(".");

    return 0;

}



Задача 3  
 Напишіть програму, яка друкує рядки з файлу, що містять слово, передане як аргумент програми (проста версія утиліти grep в UNIX).

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX\_LINE\_LENGTH 1024

void search\_in\_file(const char \*filename, const char \*word) {

    FILE \*file = fopen(filename, "r");

    if (!file) {

        printf("file not found");

        return;

    }

    char line[MAX\_LINE\_LENGTH];

    while (fgets(line, sizeof(line), file)) {

        if (strstr(line, word)) {

            printf("%s", line);

        }

    }

    fclose(file);

}

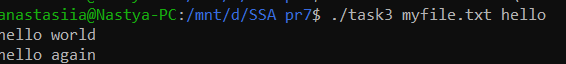
int main(int argc, char \*argv[]) {

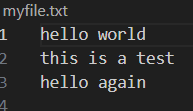
    if (argc != 3) {return 1;}

    search\_in\_file(argv[1], argv[2]);

    return 0;

}





Задача 4  
 Напишіть програму, яка виводить список файлів, заданих у вигляді аргументів, з зупинкою кожні 20 рядків, доки не буде натиснута клавіша (спрощена версія утиліти more в UNIX).

#include <stdio.h>

#include <stdlib.h>

#define LINES\_PER\_PAGE 20

void print\_file\_with\_pause(const char \*filename) {

    FILE \*file = fopen(filename, "r");

    if (!file) {

        printf("Не вдалося відкрити файл: %s\n", filename);

        return;

    }

    char line[512];

    int line\_count = 0;

    while (fgets(line, sizeof(line), file)) {

        printf("%s", line);

        line\_count++;

        if (line\_count >= LINES\_PER\_PAGE) {

            printf("\nenter ENTER to cont");

            getchar();

            line\_count = 0;

        }

    }

    fclose(file);

}

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

    if (argc < 2) {

        return 1;

    }

    for (int i = 1; i < argc; i++) {

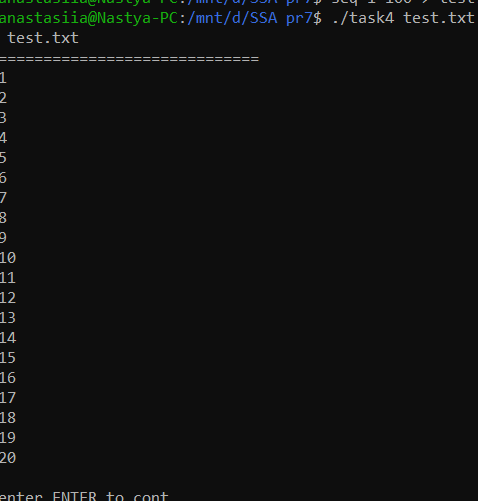
        printf(" %s\n%s\n", argv[i], "=============================");

        print\_file\_with\_pause(argv[i]);

    }

    return 0;

}



Задача 5  
 Напишіть програму, яка перелічує всі файли в поточному каталозі та всі файли в підкаталогах.

#include <stdio.h>

#include <stdlib.h>

#include <dirent.h>

#include <sys/types.h>

#include <string.h>

#include <sys/stat.h>

#include <unistd.h>

void list\_files(const char \*path) {

    struct dirent \*entry;

    DIR \*dir = opendir(path);

    if (!dir) {

        perror("cannot find catalog");

        return;

    }

    while ((entry = readdir(dir)) != NULL) {

        char full\_path[1024];

        snprintf(full\_path, sizeof(full\_path), "%s/%s", path, entry->d\_name);

        if (entry->d\_type == DT\_DIR) {

            if (strcmp(entry->d\_name, ".") == 0 ||

        strcmp(entry->d\_name, "..") == 0)

                continue;

            printf("catalog -> %s\n", full\_path);

            list\_files(full\_path);

        } else {

            printf("file -> %s\n", full\_path);

        }

    }

    closedir(dir);

}

int main() {

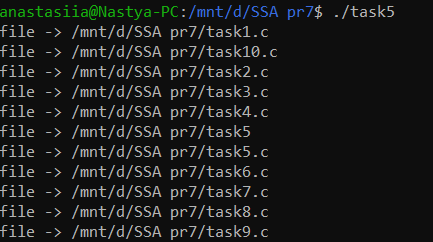
    char path[1024];

    getcwd(path, sizeof(path));

    list\_files(path);

    return 0;

}



Задача 6  
 Напишіть програму, яка перелічує лише підкаталоги у алфавітному порядку.

#include <stdio.h>

#include <stdlib.h>

#include <dirent.h>

#include <string.h>

#include <sys/stat.h>

#define MAX\_DIRS 1000

void list\_sorted\_dirs(const char \*path) {

    struct dirent \*entry;

    DIR \*dir = opendir(path);

    if (!dir) {

        perror("cannot open catalog");

        return;

    }

    char directories[MAX\_DIRS][256];

    int count = 0;

    while ((entry = readdir(dir)) != NULL) {

        struct stat statbuf;

        char full\_path[1024];

        snprintf(full\_path, sizeof(full\_path), "%s/%s", path, entry->d\_name);

        if (stat(full\_path, &statbuf) == 0 && S\_ISDIR(statbuf.st\_mode)) {

            if (strcmp(entry->d\_name, ".") != 0 && strcmp(entry->d\_name, "..") != 0) {

                strncpy(directories[count++], entry->d\_name, sizeof(directories[count]) - 1);

            }

        }

    }

    closedir(dir);

    qsort(directories, count, sizeof(directories[0]), (int (\*)(const void \*, const void \*)) strcmp);

    printf("SORTED CATALOGES in%s\n", path);

    for (int i = 0; i < count; i++) {

        printf("%s\n", directories[i]);

    }

}

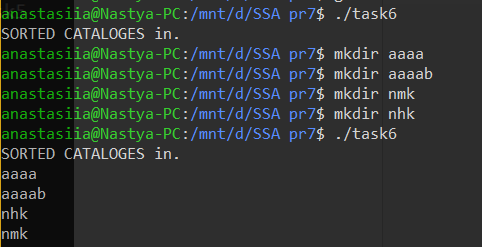
int main() {

    char path[1024] = ".";

    list\_sorted\_dirs(path);

    return 0;

}



Задача 7  
 Напишіть програму, яка показує користувачу всі його/її вихідні програми на C, а потім в інтерактивному режимі запитує, чи потрібно надати іншим дозвіл на читання (read permission); у разі ствердної відповіді — такий дозвіл повинен бути наданий.

#include <stdio.h>

#include <stdlib.h>

#include <dirent.h>

#include <sys/stat.h>

#include <string.h>

#define MAX\_FILES 100

void change\_permissions(const char \*filename) {

    if (chmod(filename, S\_IRUSR | S\_IRGRP | S\_IROTH) == 0) {

        printf("permission was given -> %s\n", filename);

    } else {

        perror("couldn't give permission");

    }

}

void list\_and\_modify\_c\_executables() {

    struct dirent \*entry;

    DIR \*dir = opendir(".");

    char files[MAX\_FILES][256];

    int count = 0;

    if (!dir) {

        perror("couldn't open catalog");

        return;

    }

    while ((entry = readdir(dir)) != NULL) {

        struct stat statbuf;

        if (stat(entry->d\_name, &statbuf) == 0 && (strstr(entry->d\_name, ".c") || strstr(entry->d\_name, ".exe"))) {

            strncpy(files[count++], entry->d\_name, sizeof(files[count]) - 1);

        }

    }

    closedir(dir);

    if (count == 0) {

        printf("out files was not found\n");

        return;

    }

    printf("\n%d files were found\n", count);

    for (int i = 0; i < count; i++) {

        printf("[%d] %s\n", i + 1, files[i]);

    }

    char response;

    printf("\nDo you want to give permission?(y/n): ");

    scanf(" %c", &response);

    if (response == 'y' || response == 'Y') {

        for (int i = 0; i < count; i++) {

            change\_permissions(files[i]);

        }

    } else {

        printf("permissions was not changed\n");

    }

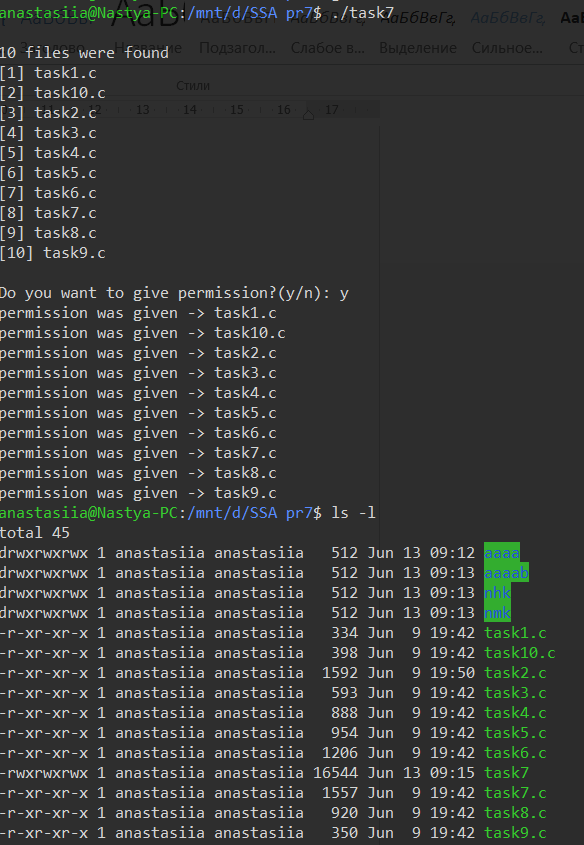
}

int main() {

    list\_and\_modify\_c\_executables();

    return 0;

}



Задача 8  
 Напишіть програму, яка надає користувачу можливість видалити будь-який або всі файли у поточному робочому каталозі. Має з’являтися ім’я файлу з запитом, чи слід його видалити.

#include <stdio.h>

#include <stdlib.h>

#include <dirent.h>

#include <string.h>

#include <unistd.h>

void delete\_file(const char \*filename) {

    if (remove(filename) == 0) {

        printf("File \"%s\" deleted successfully.\n", filename);

    } else {

        perror("Error deleting file");

    }

}

void list\_and\_delete\_files() {

    struct dirent \*entry;

    DIR \*dir = opendir(".");

    if (!dir) {

        perror("Failed to open directory");

        return;

    }

    char response;

    while ((entry = readdir(dir)) != NULL) {

        if (entry->d\_type == DT\_REG) {

            printf("Delete file \"%s\"? (y/n): ", entry->d\_name);

            scanf(" %c", &response);

            if (response == 'y' || response == 'Y') {

                delete\_file(entry->d\_name);

            }

        }

    }

    closedir(dir);

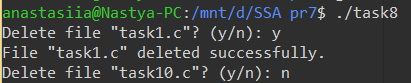
}

int main() {

    list\_and\_delete\_files();

    return 0;

}



Задача 9  
 Напишіть програму на C, яка вимірює час виконання фрагмента коду в мілісекундах.

#include <stdio.h>

#include <time.h>

int main() {

    clock\_t start, end;

    double time\_elapsed;

    start = clock();

    for (volatile int i = 0; i < 1000000; i++);

    end = clock();

    time\_elapsed = ((double)(end - start) / CLOCKS\_PER\_SEC) \* 1000.0;

    printf("Execution time: %.3f ms\n", time\_elapsed);

    return 0;

}



Задача 10  
 Напишіть програму мовою C для створення послідовності випадкових чисел з плаваючою комою у діапазонах:  
 (a) від 0.0 до 1.0  
 (b) від 0.0 до n, де n — будь-яке дійсне число з плаваючою точкою.  
 Початкове значення генератора випадкових чисел має бути встановлене так, щоб гарантувати унікальну послідовність.

Примітка: використання прапорця -Wall під час компіляції є обов’язковим.

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

double random\_0\_1() {

    return (double)rand() / RAND\_MAX;

}

double random\_0\_n(double n) {

    return random\_0\_1() \* n;

}

int main() {

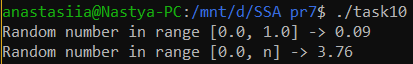
    srand(time(NULL));

    printf("Random number in range [0.0, 1.0] -> %.2f\n", random\_0\_1());

    printf("Random number in range [0.0, n] -> %.2f\n", random\_0\_n(5.0));

    return 0;

}



Напишіть утиліту, яка аналізує "заплутаність" структури директорій за спеціальним критерієм вашого вибору.

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <dirent.h>

#include <sys/stat.h>

#include <unistd.h>

#include <limits.h>

#define MAX\_NAME 256

int max\_depth = 0;

int total\_dirs = 0;

int total\_symlinks = 0;

typedef struct NameNode {

    char name[MAX\_NAME];

    int count;

    struct NameNode \*next;

} NameNode;

NameNode \*head = NULL;

void add\_or\_increment(const char \*name) {

    NameNode \*curr = head;

    while (curr) {

        if (strcmp(curr->name, name) == 0) {

            curr->count++;

            return;

        }

        curr = curr->next;

    }

    NameNode \*new\_node = malloc(sizeof(NameNode));

    strcpy(new\_node->name, name);

    new\_node->count = 1;

    new\_node->next = head;

    head = new\_node;

}

int count\_repeats() {

    int repeat\_penalty = 0;

    NameNode \*curr = head;

    while (curr) {

        if (curr->count > 1) {

            repeat\_penalty += curr->count - 1;

        }

        curr = curr->next;

    }

    return repeat\_penalty;

}

void free\_list() {

    NameNode \*curr = head;

    while (curr) {

        NameNode \*tmp = curr;

        curr = curr->next;

        free(tmp);

    }

}

void analyze\_dir(const char \*path, int depth) {

    if (depth > max\_depth) max\_depth = depth;

    DIR \*dir = opendir(path);

    if (!dir) return;

    struct dirent \*entry;

    while ((entry = readdir(dir))) {

        if (strcmp(entry->d\_name, ".") == 0 || strcmp(entry->d\_name, "..") == 0) continue;

        char full\_path[PATH\_MAX];

        snprintf(full\_path, sizeof(full\_path), "%s/%s", path, entry->d\_name);

        struct stat st;

        lstat(full\_path, &st);

        if (S\_ISDIR(st.st\_mode)) {

            total\_dirs++;

            add\_or\_increment(entry->d\_name);

            analyze\_dir(full\_path, depth + 1);

        } else if (S\_ISLNK(st.st\_mode)) {

            total\_symlinks++;

        }

    }

    closedir(dir);

}

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

    if (argc != 2) {

        printf("Використання: %s <шлях\_до\_каталогу>\n", argv[0]);

        return 1;

    }

    analyze\_dir(argv[1], 1);

    int penalty = count\_repeats();

    int complexity = (max\_depth \* 2) + total\_dirs + (total\_symlinks \* 3) + (penalty \* 2);

    printf("Максимальна глибина: %d\n", max\_depth);

    printf("Кількість директорій: %d\n", total\_dirs);

    printf("Кількість symlink: %d\n", total\_symlinks);

    printf("Штраф за повтори: %d\n", penalty);

    printf("Складність структури: %d\n", complexity);

    free\_list();

    return 0;

}

