Advanced Unix Programming
Lab 6

Purva Tendulkar : 111403049

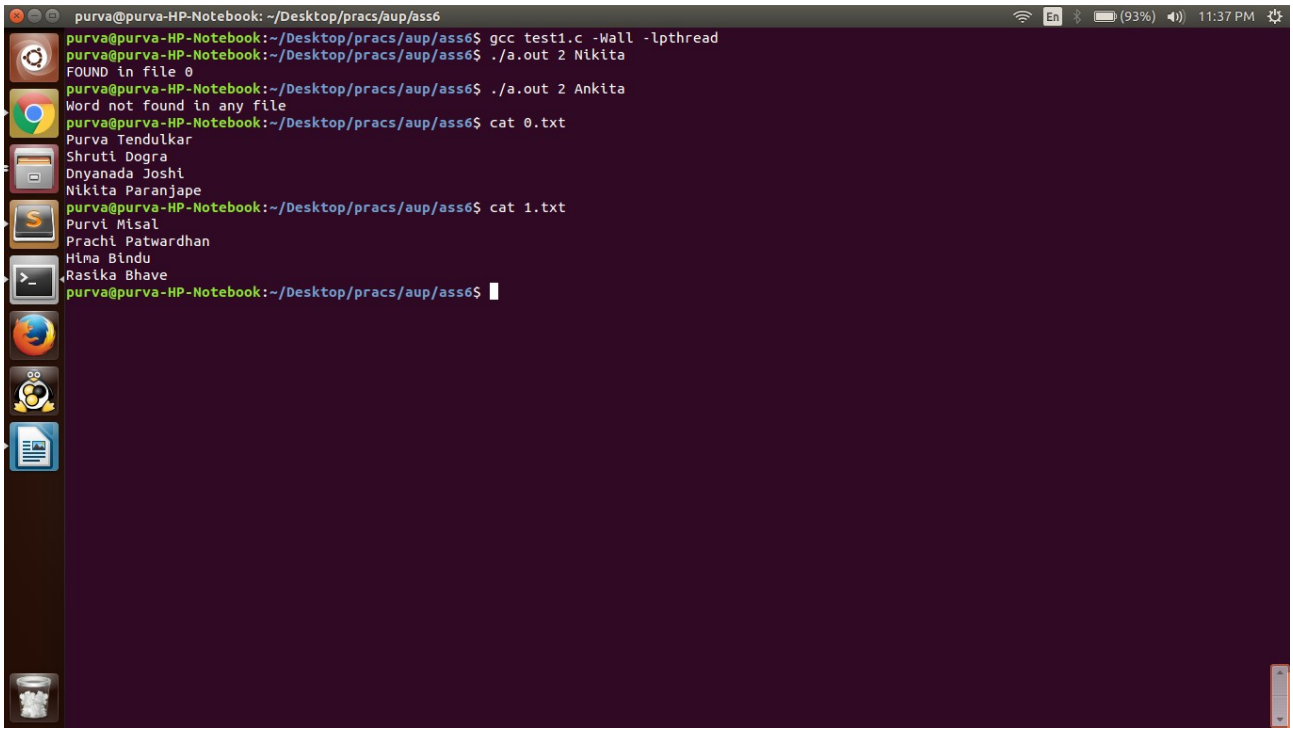
Q1. Write a program to take input from user for number of files to be scanned and word to be searched. Write a multi threaded program to search the files and return pattern if found.

Code :

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <pthread.h>

#define MAX 20

typedef struct arg_struct {
   int index;
   char word[MAX];
}arg_struct;

int set = 0;

void *myThreadFun(void *arguments) {
        struct arg_struct *args = (arg_struct *) arguments;
   int a = args->index, len_string;
   FILE *fp;
   char filename[MAX], SearchText[MAX], *buffer, *x;
   strcpy(SearchText, args->word);

   /* Opening file */
   sprintf(filename, "%d", a);
   strcat(filename, ".txt");
   fp = fopen(filename, "r");

   /* Searching logic */
        if (fp) {
                fseek (fp, 0, SEEK_END);
                len_string = ftell(fp);
                fseek(fp, 0, SEEK_SET);
                buffer = (char *)malloc(len_string);
                if (buffer)
                        fread (buffer, 1, len_string, fp);
                fclose (fp);
        }

        x = strstr(buffer, SearchText);          // finds first occurence of SearchText in buffer
        if (x != NULL) {
                printf("FOUND in file %d\n", a);
                set = 1;
```

```
                exit(0);
        }

    return NULL;
}

int main(int argc, char *argv[]) {
        int i, file_num;
        char word[20];

        if (argc != 3) {
                printf("Invalid number of arguments!\n");
                return -1;
        }

        file_num = atoi(argv[1]);
        strcpy(word, argv[2]);

    pthread_t *tid = (pthread_t *)malloc(sizeof(pthread_t) * file_num);

    /* Threads */
    for (i = 0; i < file_num; i++) {
        arg_struct *args = (arg_struct *)malloc(sizeof(arg_struct));
        args->index = i;
        strcpy(args->word, word);

        pthread_create(&tid[i], NULL, myThreadFun, (void *)args);
    }

    for (i = 0; i < file_num; i++)
        pthread_join(tid[i], NULL);

    if (!set)
        printf("Word not found in any file\n");

    exit(0);
}
```

Input & Output Screenshots :



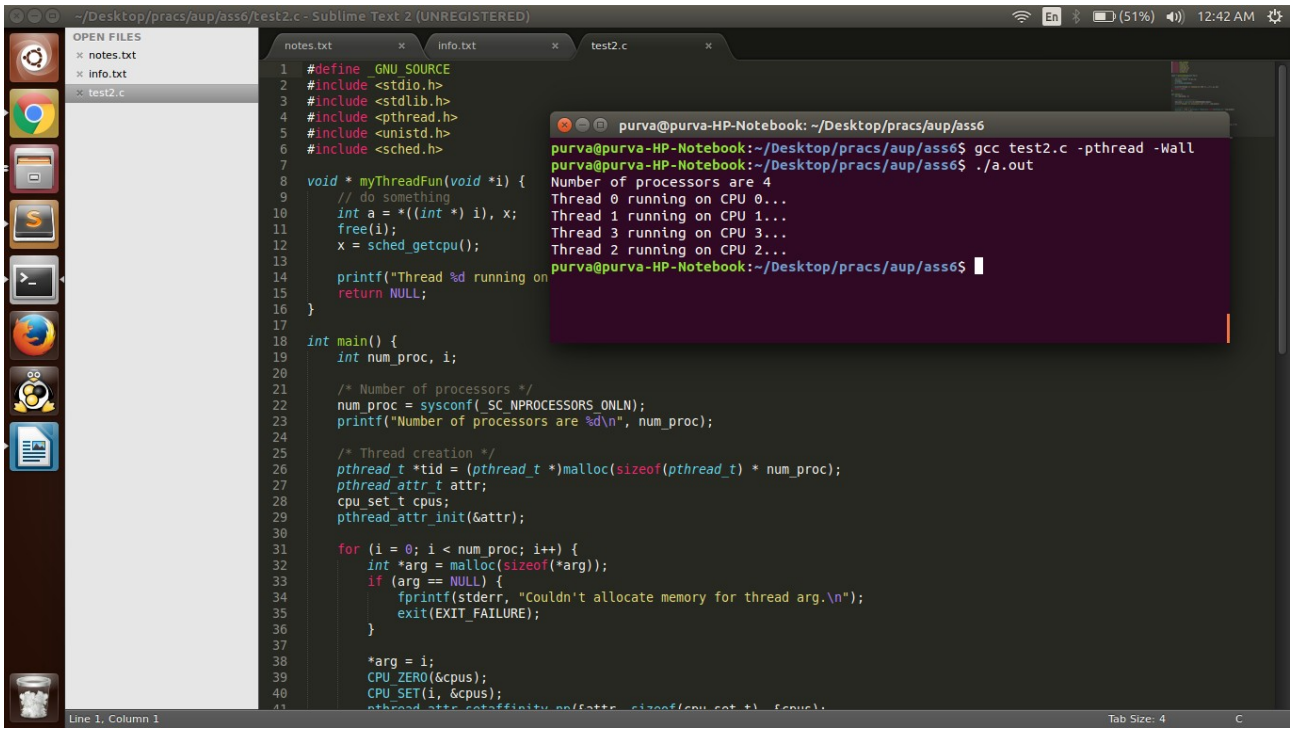
Q2. Write a program to find number of CPUs, create that many threads and attach those threads to CPUs.

Code :

```
#define _GNU_SOURCE
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <unistd.h>
#include <sched.h>

void * myThreadFun(void *i) {
        // do something
        int a = *((int *) i), x;
   free(i);
   x = sched_getcpu();

   printf("Thread %d running on CPU %d...\n", a, x);
   return NULL;
}

int main() {
        int num_proc, i;

        /* Number of processors */
        num_proc = sysconf(_SC_NPROCESSORS_ONLN);
        printf("Number of processors are %d\n", num_proc);

        /* Thread creation */
```

```
    pthread_t *tid = (pthread_t *)malloc(sizeof(pthread_t) * num_proc);
    pthread_attr_t attr;
    cpu_set_t cpus;
    pthread_attr_init(&attr);

    for (i = 0; i < num_proc; i++) {
        int *arg = malloc(sizeof(*arg));
        if (arg == NULL) {
            fprintf(stderr, "Couldn't allocate memory for thread arg.\n");
            exit(EXIT_FAILURE);
        }

        *arg = i;
        CPU_ZERO(&cpus);
        CPU_SET(i, &cpus);
        pthread_attr_setaffinity_np(&attr, sizeof(cpu_set_t), &cpus);
        pthread_create(&tid[i], &attr, myThreadFun, arg);
    }

    for (i = 0; i < num_proc; i++)
        pthread_join(tid[i], NULL);

    return 0;
}
```

Input & Output Screenshots :



Q3. Write a short program that creates 5 threads which print a tread "id" that is passed to thread function by pointer.

Code :

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <pthread.h>

#define MAX 20

typedef struct arg_struct {
   int index;
}arg_struct;

int set = 0;

void *myThreadFun(void *arguments) {
        struct arg_struct *args = (arg_struct *) arguments;
   int a = args->index;

   printf("Thread ID : %d\n", a);

   return NULL;
}

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

   pthread_t *tid = (pthread_t *)malloc(sizeof(pthread_t) * 5);

   /* Threads */
   for (i = 0; i < 5; i++) {
        arg_struct *args = (arg_struct *)malloc(sizeof(arg_struct));
        args->index = i;

        pthread_create(&tid[i], NULL, myThreadFun, (void *)args);
   }

   for (i = 0; i < 5; i++)
        pthread_join(tid[i], NULL);

   return 0;
}
```

Input & Output Screenshots :

~/Desktop/pracs/aup/ass6/test3.c - Sublime Text 2 (UNREGISTERED)

OPEN FILES
- notes.txt
- info.txt
- test2.c
- test3.c

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

#define MAX     20

typedef struct arg_struct {
    int index;
}arg_struct;

int set = 0;

void *myThreadFun(void *arguments) {
    struct arg_struct *args = (arg_struct *) arguments;
    int a = args->index;

    printf("Thread ID : %d\n", a);

    return NULL;
}

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

    pthread_t *tid = (pthread_t *)malloc(sizeof(pthread_t) * 5);

    /* Threads */
    for (i = 0; i < 5; i++) {
        arg_struct *args = (arg_struct *)malloc(sizeof(arg_struct));
        args->index = i;

        pthread_create(&tid[i], NULL, myThreadFun, (void *)args);
    }

    for (i = 0; i < 5; i++)
        pthread_join(tid[i], NULL);

    return 0;
}
```

purva@purva-HP-Notebook: ~/Desktop/pracs/aup/ass6

```
purva@purva-HP-Notebook:~/Desktop/pracs/aup/ass6$ gcc test3.c -pthread
purva@purva-HP-Notebook:~/Desktop/pracs/aup/ass6$ gcc test3.c -pthread -Wall
purva@purva-HP-Notebook:~/Desktop/pracs/aup/ass6$ ./a.out
Thread ID : 1
Thread ID : 4
Thread ID : 2
Thread ID : 0
Thread ID : 3
purva@purva-HP-Notebook:~/Desktop/pracs/aup/ass6$
```

Line 41, Column 2 Spaces: 4 C