

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 <unistd.h>
#include <pthread.h>
#include <stdlib.h>
#include <string.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <fcntl.h>
#define MAXLENGTH 32

void *search(void *arg);
int match(char *word, int fd);

// Structure helpful in searching
typedef struct {
    char *word;
    char *filename;
} pthreadstruct;

// global variable for thread exit
int flag = 0;

int main(int argc, char *argv[]) {
    if(argc < 2) {
        printf("Insufficient arguments\n");
        return -1;
    }
    char *word = argv[1];                // pattern
    int num_files = argc - 2;            // number of files
    pthreadstruct pths[num_files];       // struct for files
    pthread_t threads[num_files];        // pthread array
    int i = 0;
    for(i = 0; i < num_files; i++) {
        pths[i].word = word;
        pths[i].filename = argv[i + 2];

        if(pthread_create(&threads[i], NULL, search, &pths[i])) {
            printf("Thread creation failed\n");
            return -1;
        }
    }
    for(i = 0; i < num_files; i++) {
        void *ret;
        if(pthread_join(threads[i], &ret)) {
            printf("Thread join failed\n");
        }
    }
}
```

```

        return -1;
    }
}
return 0;
}

void *search(void *arg) {
    pthreadstruct *args = (pthreadstruct *)arg;
    int fd;
    fd = open(args->filename, O_RDONLY);
    if(fd == -1) {
        printf("File Not Found\n");
        pthread_exit(NULL);
    }
    if(match(args->word, fd)) {
        printf("%s\n", args->filename);
        flag = 1;
        pthread_exit((char*)(args->filename));
    }
    else {
        printf("Match not found in file %s\n", args->filename);
        pthread_exit(NULL);
    }
}

// utility for search function

int match(char *word, int fd) {
    char line[MAXLENGTH];
    char c;
    int i, n;
    while(((n = read(fd, line, MAXLENGTH)) > 0)) {
        if(strstr(line, word) != NULL) {
            printf("The pattern is %s\n", line);
            return 1;
        }
    }
    return 0;
}

```

Execution:

```
nikunj@nikunj-Inspiron-3543 ~/Desktop/AUP-ASSGN/LAB-6
File Edit View Search Terminal Help
nikunj@nikunj-Inspiron-3543 ~/Desktop/AUP-ASSGN/LAB-6 $ cc q1.c -o q1 -lpthread
nikunj@nikunj-Inspiron-3543 ~/Desktop/AUP-ASSGN/LAB-6 $ ./q1 printf 1.c 2.c
The pattern is argc, char *argv[] {
    printf("
1.c
The pattern is &i);
    printf("i is %d\n", i);
    r
2.c
nikunj@nikunj-Inspiron-3543 ~/Desktop/AUP-ASSGN/LAB-6 $ ./q1 scanf 1.c 2.c
The pattern is *argv[] {
    int i;
    scanf("%d",
2.c
Match not found in file 1.c
nikunj@nikunj-Inspiron-3543 ~/Desktop/AUP-ASSGN/LAB-6 $ ./q1 abcd 1.c 2.c
Match not found in file 2.c
Match not found in file 1.c
nikunj@nikunj-Inspiron-3543 ~/Desktop/AUP-ASSGN/LAB-6 $ cat 1.c
#include <stdio.h>

int main(int argc, char *argv[]) {
    printf("This is a sample input file\n");
    return 0;
}
nikunj@nikunj-Inspiron-3543 ~/Desktop/AUP-ASSGN/LAB-6 $ cat 2.c
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]) {
    int i;
    scanf("%d", &i);
    printf("i is %d\n", i);
    return 0;
}
nikunj@nikunj-Inspiron-3543 ~/Desktop/AUP-ASSGN/LAB-6 $ ./q1 printf 2.c 1.c
The pattern is &i);
    printf("i is %d\n", i);
    r
2.c
The pattern is argc, char *argv[] {
    printf("
1.c
nikunj@nikunj-Inspiron-3543 ~/Desktop/AUP-ASSGN/LAB-6 $
```

Here a struct pthread_t is declared which has char *word and char *filename as its items. An array of pthread_t is created of size argc - 2. The &pths[i] is an argument to the search function of the ith thread created. The match function acts as a utility function to search function. In the above execution "printf" and "scanf" are the words to be searched for and 1.c and 2.c are the test files containing some code which are having these keywords. If in case a match is not found it prints "Match not found in file filename".

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 <unistd.h>
#include <pthread.h>
#include <sched.h>
#include <stdlib.h>
```

```
void *threadcpu(void *arg);
int main() {
    int i, *args;
    long cpu;
    cpu_set_t *cpuset;
    pthread_t *pthreadset;
    pthread_attr_t *pthreadattrset;
```

```

cpu = sysconf(_SC_NPROCESSORS_ONLN);
printf("The system has %ld processing cores\n", cpu);

cpuset = (cpu_set_t*)malloc(sizeof(cpu_set_t) * (int)cpu);
pthreadset = (pthread_t*)malloc(sizeof(pthread_t) * (int)cpu);
pthreadattrset = (pthread_attr_t*)malloc(sizeof(pthread_attr_t) * (int)cpu);
args = (int *)malloc(sizeof(int) * (int)cpu);

for(i = 0; i < (int)cpu; i++) {
    CPU_ZERO(&cpuset[i]);
}
for(i = 0; i < (int)cpu; i++) {
    CPU_SET(i, &cpuset[i]);
}

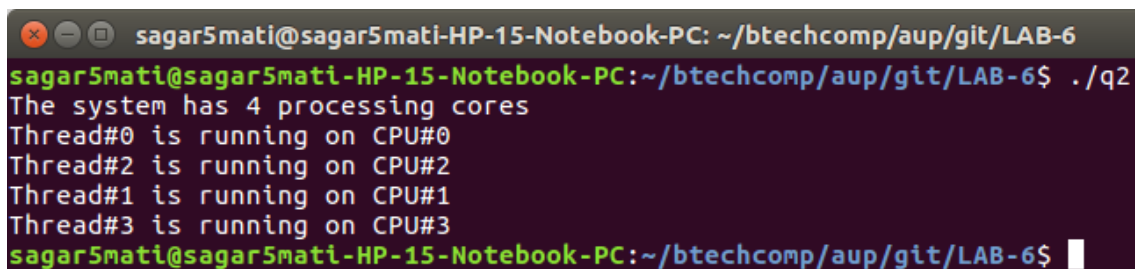
for(i = 0; i < (int)cpu; i++) {
    pthread_attr_init(&pthreadattrset[i]);
}
for(i = 0; i < (int)cpu; i++) {
    pthread_attr_setaffinity_np(&pthreadattrset[i], sizeof(cpu_set_t), &cpuset[i]);
}
for(i = 0; i < (int)cpu; i++) {
    args[i] = i;
    pthread_create(&pthreadset[i], &pthreadattrset[i], &threadcpu, (void*)&args[i]);
}

for(i = 0; i < (int)cpu; i++) {
    pthread_join(pthreadset[i], NULL);
}
return 0;
}

void *threadcpu(void *arg) {
    int argu;
    argu = *(int *)arg;
    printf("Thread#%d is running on CPU#%d\n", argu, sched_getcpu());
    return NULL;
}

```

Execution:



```

sagar5mati@sagar5mati-HP-15-Notebook-PC: ~/btechcomp/aup/git/LAB-6
sagar5mati@sagar5mati-HP-15-Notebook-PC:~/btechcomp/aup/git/LAB-6$ ./q2
The system has 4 processing cores
Thread#0 is running on CPU#0
Thread#2 is running on CPU#2
Thread#1 is running on CPU#1
Thread#3 is running on CPU#3
sagar5mati@sagar5mati-HP-15-Notebook-PC:~/btechcomp/aup/git/LAB-6$

```

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

Code:

```
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>
#include <ctype.h>
#include <unistd.h>
int charToInt(char *str) {
    int result = 0, count = 0;
    while(str[count]) {
        if(isdigit(str[count])) {
            result = result * 10 + (str[count] - '0');
        }
        else {
            return INT_MIN;
        }
        count++;
    }
    return result;
}
void *function(void *arg) {
    int *a;
    a = (int*) arg;
    printf("Thread number = %d\n", *a);
    pthread_exit(0);
}
int main(int argc, char *argv[]) {
    if(argc < 2) {
        printf("Not enough arguments\n");
        return 1;
    }
    int num, count, i;
    pthread_t *th;
    num = charToInt(argv[1]);
    th = (pthread_t *) malloc(sizeof(pthread_t) * num);
    for(count = 0; count < num; count++) {
        i = count;
        pthread_create(&th[count], NULL, function, &i);
        pthread_join(th[count], NULL);
    }
    return 0;
}
```

Execution:

```
abhijeet@abhijeet: ~/Advanced-Unix-Programming---Lab/LAB-6
abhijeet@abhijeet:~/Advanced-Unix-Programming---Lab/LAB-6$ cc 3.c -Wall -o 3 -lpthread
abhijeet@abhijeet:~/Advanced-Unix-Programming---Lab/LAB-6$ ./3 7
Thread number = 0
Thread number = 1
Thread number = 2
Thread number = 3
Thread number = 4
Thread number = 5
Thread number = 6
abhijeet@abhijeet:~/Advanced-Unix-Programming---Lab/LAB-6$ ./3 5
Thread number = 0
Thread number = 1
Thread number = 2
Thread number = 3
Thread number = 4
abhijeet@abhijeet:~/Advanced-Unix-Programming---Lab/LAB-6$ ./3 0
abhijeet@abhijeet:~/Advanced-Unix-Programming---Lab/LAB-6$ ./3 2
Thread number = 0
Thread number = 1
abhijeet@abhijeet:~/Advanced-Unix-Programming---Lab/LAB-6$ ./3 4
Thread number = 0
Thread number = 1
Thread number = 2
Thread number = 3
abhijeet@abhijeet:~/Advanced-Unix-Programming---Lab/LAB-6$
```

The above program takes the number of threads to be created a command line argument. It then passes an integer to function which prints the integer.