**File →** touch, cat, cp [filename] [directory], mv, rm, ls -l, chmod [u, g, o +- r, w, x], chmod u-w,g-rw,o+w filename, grep [-I,c,w,o,n]CSE filename.txt, sed -n 1,3p course.txt

**Array Pointer Struct**

#include <stdio.h> #include <stdio.h> starcpy(s1,s2) Copies s2 into s1

int main() { int main() { strcat(s1,s2) Concatenates s2 on the end of s1

int values[5]; int x[5] = {1, 2, 3, 4, 5}; strlen(s1) Returns length

printf("Enter 5 integers: "); int\* ptr; strcmp(s1,s2) Compares s1,s2

for(int i = 0; i < 5; ++i) { ptr = &x[2]; strchr, strstr Returns pointer of first occurrence

scanf("%d", &values[i]); } printf("\*ptr = %d \n", \*ptr); // 3 **Command Line Arguments**

printf("Displaying integers: "); return 0; int main (int argc, char \*argv[]) {

for(int i = 0; i < 5; ++i) { } printf("argc: %d\n", argc);

printf("%d\n", values[i]); } printf("argv[0],%s\n", argv[0]);

return 0; } return 0; }

**1. gedit file.c 2. gcc-o build file.c / gcc-o build file.c -lpthread 3. ./build / ./build 1 2**

**File descriptor** 0 → input, 1 → output, 2 → error, >2 → assigned to files we open / work with

**Threads**

#include <stdio.h> int main()

#include <pthread.h> {

#include <unistd.h> pthread\_t thread1, thread2;

int arr [2] = {10, 20}; pthread\_create(&thread1, NULL, adder, (void\* )arr);

void\* adder(void\* arg) pthread\_create(&thread2, NULL, subtracter, (void\* )arr);

{ pthread\_join(thread1, NULL);

int \*index = arg; pthread\_join(thread2, NULL);

int sum = index[0] + index[1]; }

printf("I am Thread: 1 performing addition.\n") ;

sleep(1);

printf("The sum is: %d.\n", sum); **Semaphores**

} int sem\_init(sem\_t \* sem, int pshared, unsigned int value);

void\* subtracter(void\* arg) initialize semaphore \*sem, shared among all if pshared 0

{ int sem\_wait(sem\_t \* sem);

int \*index = arg; int sem\_post(sem\_t \* sem);

int diff = index[1] - index[0]; int sem\_destroy(sem\_t \* sem);

printf("I am Thread: 2 performing subtraction.\n");

printf("The difference is: %d.\n", diff);

}

**int open(file\_name, mode), int read/write(file\_descriptor, buffer\_pointer, transfer\_size), int close(file\_descriptor)**

**lseek(fd,5,SEEK\_CUR) –** this moves the pointer 5 positions ahead from the current position in the file

#include <unistd.h> **Mutex**

#include <sys/wait.h> int pthread\_mutex\_init(pthread\_mutex\_t \* restrict mutex, const pthread\_mutexattr\_t \* restrict attr);

int main() { → initializes \*mutex, if attr is NULL, a default set of attributes is used. [initialize & unlocked]

pid\_t pid = fork(); int pthread\_mutex\_destroy(pthread\_mutex\_t \* restrict mutex);

if (pid < 0) { int pthread\_mutex\_lock(pthread\_mutex\_t \* mutex);

perror("Fork failed"); int pthread\_mutex\_unlock(pthread\_mutex\_t \* mutex);

exit(EXIT\_FAILURE);

} else if (pid == 0) {

int sum\_even = 0;

for (int i = 2; i <= 100; i += 2) {

sum\_even += i;}

printf("Child process (PID: %d) - Sum of even: %d\n", getpid(), sum\_even);

exit(EXIT\_SUCCESS);

} else {

int sum\_odd = 0;

for (int i = 1; i <= 100; i += 2) {

sum\_odd += i;}

printf("Parent process (PID: %d) - Sum of odd: %d\n", getpid(), sum\_odd);

int status;

waitpid(pid, &status, 0);

printf("Parent process (PID: %d) - Child process (PID: %d) has completed.\n", getpid(), pid);}

**return 0;}**

**#include <pthread.h>** pthread\_mutex\_destroy(&mutex);

int t\_id[2]={1,2}; printf("Total count: %d\n",count);

void \*t\_func(int \*id); return 0;

int count=0; }

pthread\_mutex\_t mutex; void \*t\_func(int \*id){

int main(){ printf("Entered in Thread %d...\n",\*id);

pthread\_t t[2]; for(int i=0;i<100000;i++){

pthread\_mutex\_init(&mutex,NULL); pthread\_mutex\_lock(&mutex);

pthread\_create(&t[0],NULL,(void \*)t\_func,&t\_id[0]); count++;

pthread\_create(&t[1],NULL,(void \*)t\_func,&t\_id[1]); pthread\_mutex\_unlock(&mutex);

**for(int i=0;i<2;i++){ }**

pthread\_join(t[i],NULL);} }