

```
1 // 5.1.a
2
3 #include <stdio.h>
4 #include <stdlib.h>
5 #include <errno.h>
6 #include <unistd.h>
7 #include <sys/wait.h>
8
9 #define N 1000
10
11 int main(){
12     pid_t pid;
13
14     for(int i=0; i<N; i++){
15         pid=fork();
16         if(pid==0){exit(0);}
17         else if(pid==-1){
18             printf("Failed to create process %d\n",i);
19             perror("fork"); exit(errno);
20         }
21     }
22
23     while(wait(NULL)!=-1);
24
25     return 0;
26 }
27
```

```
1 // 5.1.b
2
3 #include <stdio.h>
4 #include <stdlib.h>
5 #include <pthread.h>
6
7 #define N 1000
8
9 void *thread_func(void *arg){pthread_exit(NULL);}
10
11 int main(){
12     int ret;
13     pthread_t tids[N];
14
15     for(int i=0; i<N; i++){
16         ret = pthread_create(&tids[i], NULL, thread_func, NULL);
17         if(ret!=0){
18             printf("Failed to create thread %d (error %d)\n", i, ret);
19             exit(ret);
20         }
21     }
22
23     pthread_exit(NULL);
24 }
25
```

```
1 // 5.2.a
2
3 #include <stdio.h>
4 #include <pthread.h>
5
6 void *thread_func(void *arg){
7     long num = (long)arg;
8     printf("%ld\n", num);
9     pthread_exit(NULL);
10 }
11
12 int main(){
13     long i;
14     pthread_t tids[3];
15     for(i=0; i<3; i++)
16         pthread_create(&tids[i], NULL, thread_func, (void*)i);
17     pthread_exit(NULL);
18 }
19
```

```
1 // 5.2.b
2
3 #include <stdio.h>
4 #include <stdlib.h>
5 #include <pthread.h>
6
7 void *thread_func(void *arg){
8     int *nums = (int *)arg;
9     printf("%d\t%d\n", nums[0], nums[1]);
10    free(arg);
11    pthread_exit(NULL);
12 }
13
14 int main(){
15     int i, *nums;
16     pthread_t tids[3];
17     for(i=0; i<3; i++) {
18         nums = (int *)malloc(2*sizeof(int));
19         nums[0] = i; nums[1] = i+1;
20         pthread_create(&tids[i], NULL, thread_func, nums);
21     }
22     pthread_exit(NULL);
23 }
24
```

```
1 // 5.2.c
2
3 #include <stdio.h>
4 #include <stdlib.h>
5 #include <pthread.h>
6
7 typedef struct{float f; char c;} thread_param;
8
9 void *thread_func(void *arg){
10     thread_param *param = (thread_param *)arg;
11     printf("%f\t%c\n", param->f, param->c);
12     free(arg);
13     pthread_exit(NULL);
14 }
15
16 int main(){
17     pthread_t tids[3];
18
19     for(int i=0; i<3; i++){
20         thread_param *arg = (thread_param *)malloc(sizeof(thread_param));
21         arg->f=(float)i;
22         arg->c='a'+(int)i;
23         pthread_create(&tids[i], NULL, thread_func, arg);
24     }
25
26     pthread_exit(NULL);
27 }
28
```

```
1 // 5.3.a
2
3 #include <stdio.h>
4 #include <stdlib.h>
5 #include <pthread.h>
6
7 int sum(int min, int max){
8     int i, result = 0;
9     for(i=min; i<=max; i++) result += i;
10    return result;
11 }
12
13 int partial_sum;
14
15 void *thread_func(void *arg){
16     partial_sum = sum(51, 100);
17     pthread_exit(NULL);
18 }
19
20 int main(){
21     pthread_t tid;
22     int total_sum;
23     pthread_create(&tid, NULL, thread_func, NULL);
24     total_sum = sum(1, 50);
25     pthread_join(tid, NULL);
26     total_sum += partial_sum;
27     printf("total sum: %d\n", total_sum);
28     pthread_exit(NULL);
29 }
30
```

```
1 // 5.3.b
2
3 #include <stdio.h>
4 #include <stdlib.h>
5 #include <pthread.h>
6
7 int sum(int min, int max){
8     int i, result = 0;
9     for(i=min; i<=max; i++) result += i;
10    return result;
11 }
12
13 void *thread_func(void *arg){
14     long partial_sum = sum(51, 100);
15     pthread_exit((void *)partial_sum);
16 }
17
18 int main(){
19     pthread_t tid;
20     void *retval;
21     long total_sum;
22     pthread_create(&tid, NULL, thread_func, NULL);
23     total_sum = sum(1, 50);
24     pthread_join(tid, &retval);
25     total_sum += (long)retval;
26     printf("total sum: %ld\n", total_sum);
27     pthread_exit(NULL);
28 }
29
```

```
1 // 5.4
2
3 #include <stdio.h>
4 #include <stdlib.h>
5 #include <pthread.h>
6
7 int sum(int min, int max){
8     int i, result = 0;
9     for(i=min; i<=max; i++) result += i;
10    return result;
11 }
12
13 int partial_sum[10];
14
15 void *thread_func(void *arg){
16     long i = (long)arg;
17     partial_sum[i] = sum(i*10+1, i*10+10);
18     pthread_exit(NULL);
19 }
20
21 int main(){
22     long i;
23     pthread_t tids[10];
24     int total_sum = 0;
25
26     for(i=0; i<10; i++)
27         pthread_create(&tids[i], NULL, thread_func, (void *)i);
28
29     for(i=0; i<10; i++){
30         pthread_join(tids[i], NULL);
31         total_sum += partial_sum[i];
32     }
33
34     printf("totalsum: %d\n", total_sum);
35     return 0;
36 }
37
```



```
1 // 5.5
2
3 #include <stdio.h>
4 #include <pthread.h>
5
6 int square_sum = 0; // square_sum will be accessed sequentially, one thread at a time
7
8 void *thread_func(void *arg){
9     long i = (long)arg;
10    square_sum += i * i;
11    if(i<10) { // create next thread if n < 10
12        pthread_t tid;
13        pthread_create(&tid, NULL, thread_func, (void*)(i+1));
14        pthread_join(tid, NULL);
15    }
16    pthread_exit(NULL);
17 }
18
19 int main(){
20     long i = 1;
21     pthread_t tid;
22     pthread_create(&tid, NULL, thread_func, (void*)i); // start the chain with thread 1
23     pthread_join(tid, NULL);
24     printf("square sum: %d\n", square_sum);
25     pthread_exit(NULL);
26 }
27
```

```

1 // 5.6
2
3 #include <stdio.h>
4 #include <stdlib.h>
5 #include <unistd.h>
6 #include <pthread.h>
7
8 #define N 10
9 #define N_THREADS 2
10 #define TARGET 3
11
12 int *A;
13
14 void *thread_func(void *arg) {
15     long i = (long)arg, count = 0;
16     for(int j=i*(N/N_THREADS); j<(i+1)*(N/N_THREADS); j++)
17         if(A[j]==TARGET) count++;
18     pthread_exit((void *)count);
19 }
20
21 int main() {
22     pthread_t tids[N_THREADS];
23     void *retval;
24     long total = 0;
25
26     A = (int *)malloc(N*sizeof(int));
27     srand(getpid());
28     for (int i = 0; i < N; i++) {
29         A[i] = random() % 5;
30         printf("%d\n", A[i]);
31     }
32
33     for (long i = 0; i < N_THREADS; i++)
34         pthread_create(&tids[i], NULL, thread_func, (void *)i);
35
36     for (int i = 0; i < N_THREADS; i++) {
37         pthread_join(tids[i], &retval);
38         total += (long)retval;
39     }
40
41     printf("count=%ld\n", total);
42
43     free(A);
44     return 0;
45 }
46

```

```
1 // 5.7.a
2
3 #include <stdio.h>
4 #include <pthread.h>
5
6 unsigned long int counter=0;
7
8 void *thread_func(void *arg){
9     for(unsigned long int i=0; i<1000000; i++){
10         counter++;
11     }
12 }
13
14 int main(){
15     pthread_t tid;
16
17     pthread_create(&tid, NULL, thread_func, NULL);
18
19     thread_func(NULL);
20
21     pthread_join(tid, NULL);
22
23     printf("counter: %lu\n", counter);
24     pthread_exit(NULL);
25 }
26
```

```
1 // 5.7.b
2
3 #include <stdio.h>
4 #include <pthread.h>
5
6 unsigned long int counter=0;
7 pthread_mutex_t counter_mutex=PTHREAD_MUTEX_INITIALIZER;
8
9 void *thread_func(void *arg){
10     for(unsigned long int i=0; i<1000000; i++){
11         pthread_mutex_lock(&counter_mutex);
12         counter++;
13         pthread_mutex_unlock(&counter_mutex);
14     }
15 }
16
17 int main(){
18     pthread_t tid;
19
20     pthread_create(&tid, NULL, thread_func, NULL);
21
22     thread_func(NULL);
23
24     pthread_join(tid, NULL);
25
26     printf("counter: %lu\n", counter);
27     pthread_mutex_destroy(&counter_mutex);
28     pthread_exit(NULL);
29 }
30
```

```
1 // 5.7.c
2
3 #include <stdio.h>
4 #include <pthread.h>
5
6 unsigned long int counter=0;
7 pthread_mutex_t mutex_main=PTHREAD_MUTEX_INITIALIZER;
8 pthread_mutex_t mutex_sec=PTHREAD_MUTEX_INITIALIZER;
9
10 void *thread_func(void *arg){
11     for(unsigned long int i=0; i<1000000; i++){
12         pthread_mutex_lock(&mutex_sec);
13         counter++;
14         printf("sec: %ld\n", counter);
15         pthread_mutex_unlock(&mutex_main);
16     }
17 }
18
19 int main(){
20     pthread_t tid;
21
22     pthread_mutex_lock(&mutex_sec);
23
24     pthread_create(&tid, NULL, thread_func, NULL);
25
26     for(unsigned long int i=0; i<1000000; i++){
27         pthread_mutex_lock(&mutex_main);
28         counter++;
29         printf("main: %ld\n", counter);
30         pthread_mutex_unlock(&mutex_sec);
31     }
32
33     pthread_join(tid, NULL);
34
35     printf("counter: %lu\n", counter);
36     pthread_mutex_destroy(&mutex_main);
37     pthread_mutex_destroy(&mutex_sec);
38     pthread_exit(NULL);
39 }
40
```

```
1 // 5.8
2
3 #include <stdio.h>
4 #include <stdlib.h>
5 #include <pthread.h>
6
7 int sum(int min, int max){
8     int i, result = 0;
9     for(i=min; i<=max; i++) result += i;
10    return result;
11 }
12
13 int total_sum = 0;
14 pthread_mutex_t sum_mutex = PTHREAD_MUTEX_INITIALIZER;
15
16 void *thread_func(void *arg){
17     long i = (long)arg;
18     int partial_sum = sum(i*10+1, i*10+10);
19     pthread_mutex_lock(&sum_mutex);
20     total_sum += partial_sum;
21     pthread_mutex_unlock(&sum_mutex);
22     pthread_exit(NULL);
23 }
24
25 int main(){
26     long i;
27     pthread_t tids[10];
28
29     for(i=0; i<10; i++)
30         pthread_create(&tids[i], NULL, thread_func, (void *)i);
31
32     for(i=0; i<10; i++){
33         pthread_join(tids[i], NULL);
34     }
35
36     printf("total sum: %d\n", total_sum);
37     pthread_mutex_destroy(&sum_mutex);
38     return 0;
39 }
40
```

```

1 // 5.9
2
3 #include <stdio.h>
4 #include <stdlib.h>
5 #include <unistd.h>
6 #include <pthread.h>
7
8 #define N 10
9 #define N_THREADS 2
10 #define TARGET 3
11
12 int *A;
13 long total_count = 0;
14 pthread_mutex_t total_mutex = PTHREAD_MUTEX_INITIALIZER;
15
16 void *thread_func(void *arg){
17     long i = (long)arg, count = 0;
18     for(int j=i*(N/N_THREADS); j<(i+1)*(N/N_THREADS); j++)
19         if(A[j]==TARGET) count++;
20     pthread_mutex_lock(&total_mutex);
21     total_count += count;
22     pthread_mutex_unlock(&total_mutex);
23     pthread_exit(NULL);
24 }
25
26 int main(){
27     pthread_t tids[N_THREADS];
28
29     A = (int *)malloc(N*sizeof(int));
30     srandom(getpid());
31     for(int i = 0; i < N; i++){
32         A[i] = random() % 5;
33         printf("%d\n", A[i]);
34     }
35
36     for(long i = 0; i < N_THREADS; i++)
37         pthread_create(&tids[i], NULL, thread_func, (void *)i);
38
39     for(int i = 0; i < N_THREADS; i++)
40         pthread_join(tids[i], NULL);
41
42     printf("count=%ld\n", total_count);
43
44     free(A);
45     pthread_mutex_destroy(&total_mutex);
46     return 0;
47 }
48

```