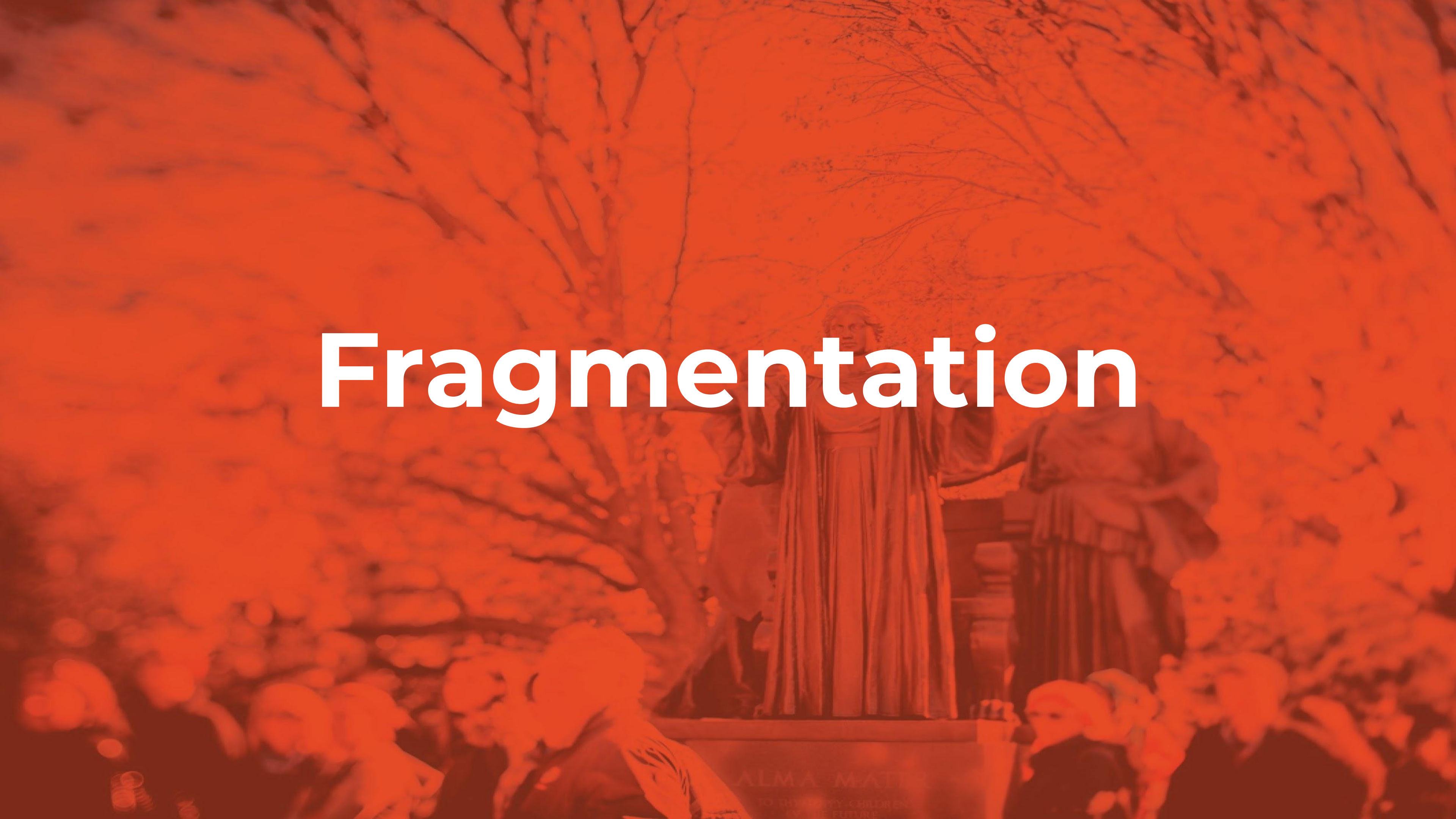
# Fragmentation, Threads, and pthreads

## CS 240 - The University of Illinois

Wade Fagen-Ulmschneider
September 9, 2021

# Fragmentation

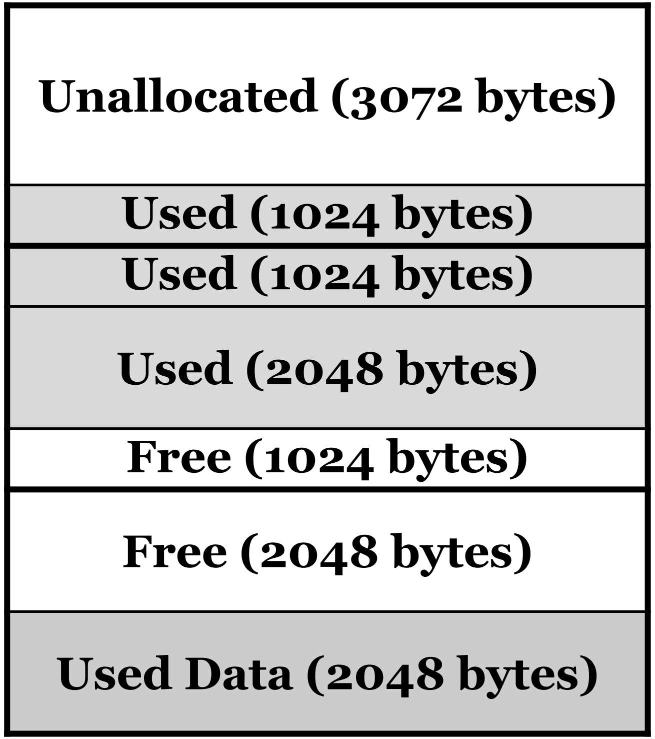


# Fragmentation

# Internal Fragmentation:

# External Fragmentation:

# Fragmentation Example



Unallocated (3072 bytes)

Used (1024 bytes)

Used (1024 bytes)

Used (2048 bytes)

Free (1024 bytes)

Free (2048 bytes)

Used Data (2048 bytes)

Start of Heap

# Threads

# Threads

# Threads

A _______________ is an organization of one or more threads in the same context.

*A simple process has only one thread.*

# Threads

In C, the initial thread is called the

______________________________.

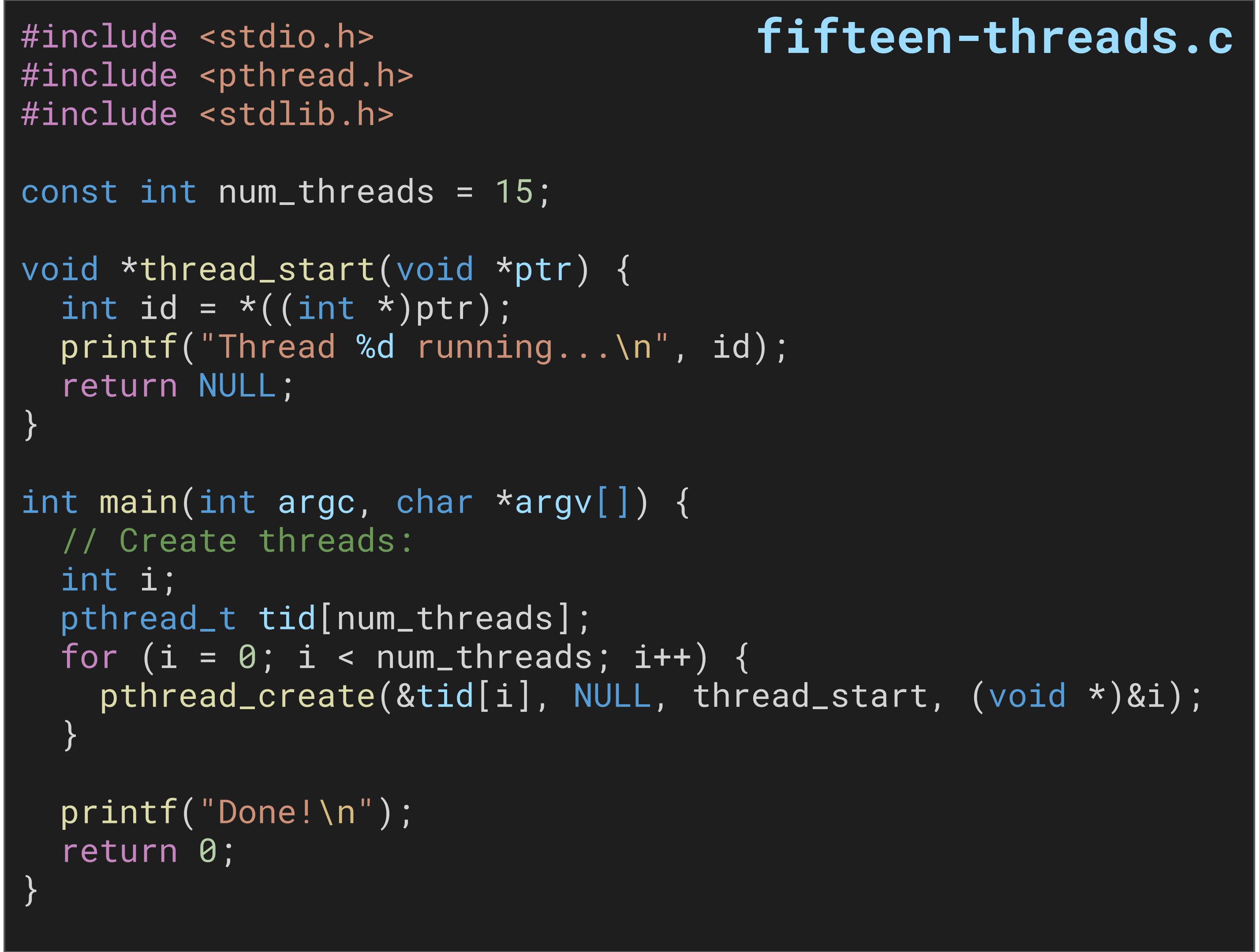
# pthread

```
int pthread_create(
  pthread_t *thread,               /* thread struct */
  const pthread_attr_t *attr,      /* usually NULL  */
  void *(*start_routine) (void *), /* start func    */
  void *arg                    /* thread start arg  */
);
```

# fifteen-threads.c

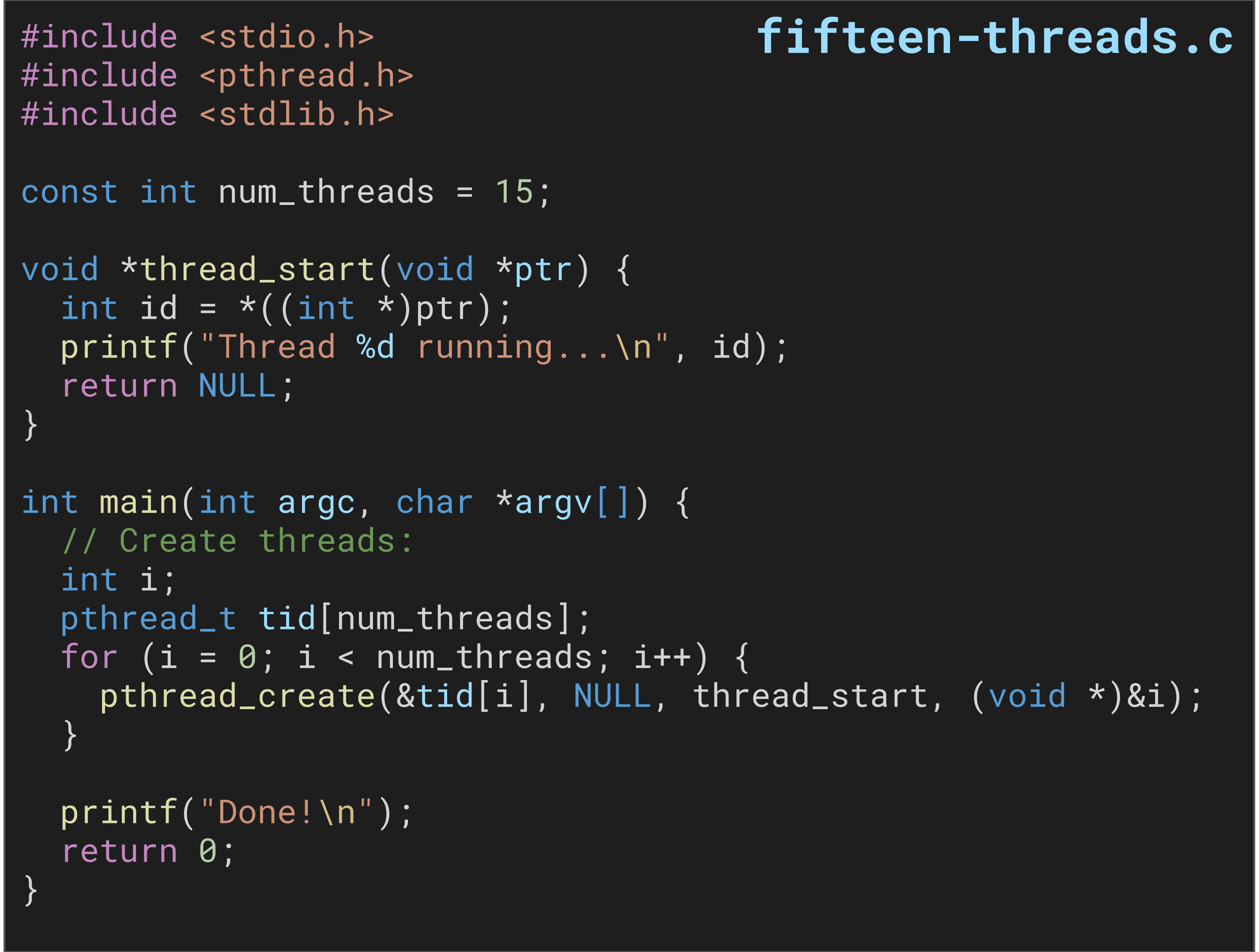
**fifteen-threads.c**

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

const int num_threads = 15;

void *thread_start(void *ptr) {
  int id = *((int *)ptr);
  printf("Thread %d running...\n", id);
  return NULL;
}

int main(int argc, char *argv[]) {
  // Create threads:
  int i;
  pthread_t tid[num_threads];
  for (i = 0; i < num_threads; i++) {
    pthread_create(&tid[i], NULL, thread_start, (void *)&i);
  }

  printf("Done!\n");
  return 0;
}

```



fifteen-threads.c

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

const int num_threads = 15;

void *thread_start(void *ptr) {
  int id = *((int *)ptr);
  printf("Thread %d running...\n", id);
  return NULL;
}

int main(int argc, char *argv[]) {
  // Create threads:
  int i;
  pthread_t tid[num_threads];
  for (i = 0; i < num_threads; i++) {
    pthread_create(&tid[i], NULL, thread_start, (void *)&i);
  }

  printf("Done!\n");
  return 0;
}

```



fifteen-threads.c

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

const int num_threads = 15;

void *thread_start(void *ptr) {
  int id = *((int *)ptr);
  printf("Thread %d running...\n", id);
  return NULL;
}

int main(int argc, char *argv[]) {
  // Create threads:
  int i;
  pthread_t tid[num_threads];
  for (i = 0; i < num_threads; i++) {
    pthread_create(&tid[i], NULL, thread_start, (void *)&i);
  }

  printf("Done!\n");
  return 0;
}

```

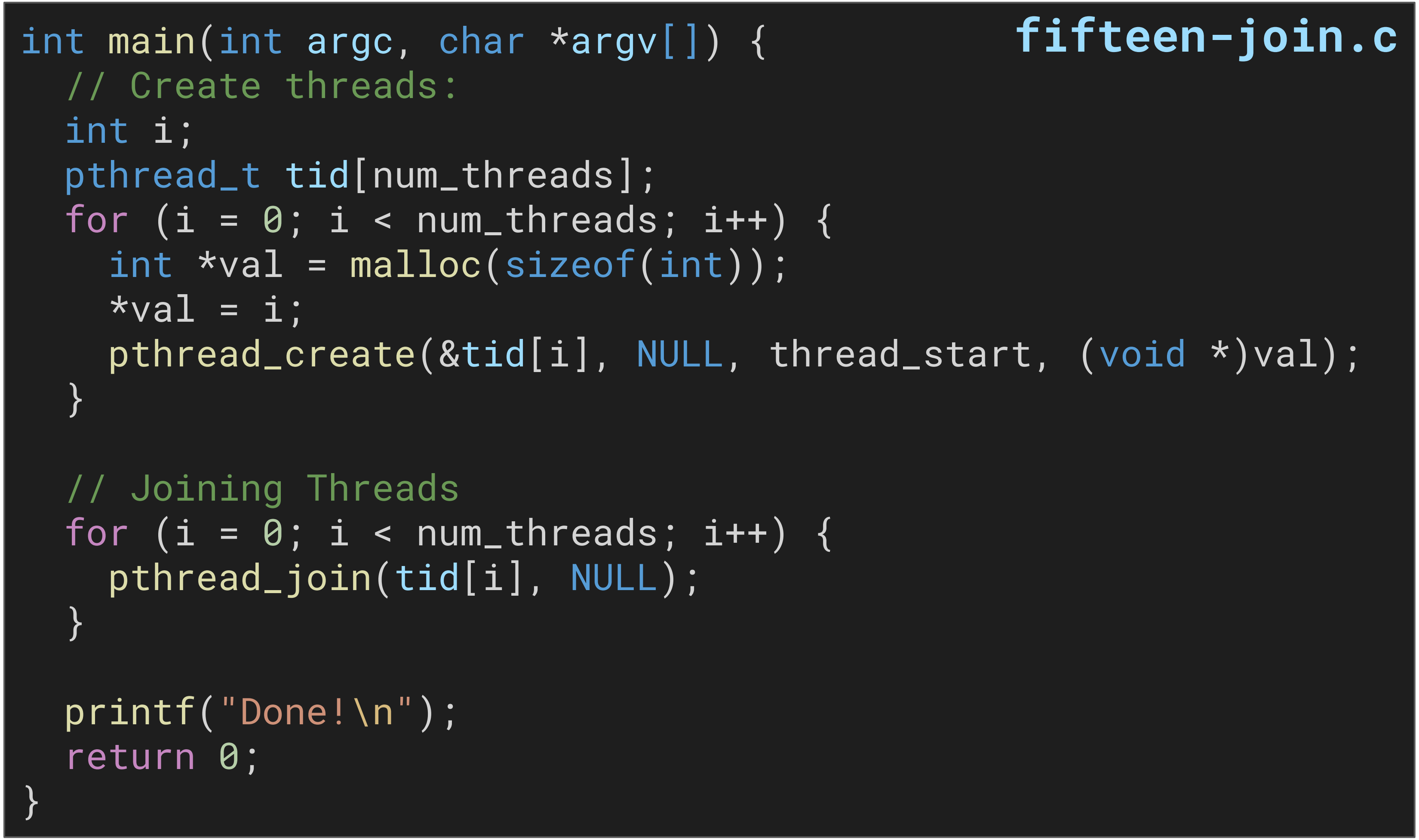
# fifteen-join.c

fifteen-join.c

```c
int main(int argc, char *argv[]) {
  // Create threads:
  int i;
  pthread_t tid[num_threads];
  for (i = 0; i < num_threads; i++) {
    int *val = malloc(sizeof(int));
    *val = i;
    pthread_create(&tid[i], NULL, thread_start, (void *)val);
  }

  // Joining Threads
  for (i = 0; i < num_threads; i++) {
    pthread_join(tid[i], NULL);
  }

  printf("Done!\n");
  return 0;
}
```

fifteen-join.c

```c
int main(int argc, char *argv[]) {
  // Create threads:
  int i;
  pthread_t tid[num_threads];
  for (i = 0; i < num_threads; i++) {
    int *val = malloc(sizeof(int));
    *val = i;
    pthread_create(&tid[i], NULL, thread_start, (void *)val);
  }

  // Joining Threads
  for (i = 0; i < num_threads; i++) {
    pthread_join(tid[i], NULL);
  }

  printf("Done!\n");
  return 0;
}
```



fifteen-join.c

```c
int main(int argc, char *argv[]) {
  // Create threads:
  int i;
  pthread_t tid[num_threads];
  for (i = 0; i < num_threads; i++) {
    int *val = malloc(sizeof(int));
    *val = i;
    pthread_create(&tid[i], NULL, thread_start, (void *)val);
  }

  // Joining Threads
  for (i = 0; i < num_threads; i++) {
    pthread_join(tid[i], NULL);
  }

  printf("Done!\n");
  return 0;
}
```