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

#define MAX_BLOCKS 10
#define MAX_FILES 5
#define MAX_FILE_NAME 20

int disk[MAX_BLOCKS];

typedef struct {
  char name[MAX_FILE_NAME];
  int start_block;
  int size;
} SequentialFile;

typedef struct {
  char name[MAX_FILE_NAME];
  int index_blocks[MAX_BLOCKS];
  int num_blocks;
} IndexedFile;

typedef struct Node {
  int block;
  struct Node *next;
} LinkedNode;

typedef struct {
  char name[MAX_FILE_NAME];
  LinkedNode *head;
} LinkedFile;

void initialize_disk() {
  for (int i = 0; i < MAX_BLOCKS; i++) {
    disk[i] = 0;
  }
}

void print_disk() {
  printf("Disk blocks: ");
  for (int i = 0; i < MAX_BLOCKS; i++) {
    printf("%d ", disk[i]);
  }
  printf("\n");
}

void allocate_sequential(SequentialFile *file, int start_block, int size) {
  if (start_block + size > MAX_BLOCKS) {
    printf("Error: Not enough space on the disk.\n");
    return;
  }
  for (int i = start_block; i < start_block + size; i++) {
    if (disk[i] != 0) {
      printf("Error: Space already allocated.\n");
      return;
    }
  }
  file->start_block = start_block;
  file->size = size;
  strcpy(file->name, "SequentialFile");
  for (int i = start_block; i < start_block + size; i++) {
    disk[i] = 1;
  }
  printf("Sequential file '%s' allocated from block %d to %d.\n", file->name,
         start_block, start_block + size - 1);
}

void allocate_indexed(IndexedFile *file, int blocks[], int num_blocks) {
  if (num_blocks > MAX_BLOCKS) {
    printf("Error: Too many blocks requested.\n");
    return;
  }
  for (int i = 0; i < num_blocks; i++) {
    if (blocks[i] >= MAX_BLOCKS || disk[blocks[i]] != 0) {
      printf("Error: Block %d is not available.\n", blocks[i]);
      return;
    }
  }
  file->num_blocks = num_blocks;
  strcpy(file->name, "IndexedFile");
  for (int i = 0; i < num_blocks; i++) {
    disk[blocks[i]] = 1; // Mark block as allocated
    file->index_blocks[i] = blocks[i];
  }
  printf("Indexed file '%s' allocated at blocks: ", file->name);
  for (int i = 0; i < num_blocks; i++) {
    printf("%d ", file->index_blocks[i]);
  }
  printf("\n");
}

void allocate_linked(LinkedFile *file, int blocks[], int num_blocks) {
  LinkedNode *prev = NULL;
  LinkedNode *head = NULL;
  for (int i = 0; i < num_blocks; i++) {
    if (blocks[i] >= MAX_BLOCKS || disk[blocks[i]] != 0) {
      printf("Error: Block %d is not available.\n", blocks[i]);
      while (head != NULL) {
        LinkedNode *temp = head;
        head = head->next;
        free(temp);
      }
      return;
    }
    disk[blocks[i]] = 1;
    LinkedNode *node = (LinkedNode *)malloc(sizeof(LinkedNode));
    node->block = blocks[i];
    node->next = NULL;
    if (prev == NULL) {
      head = node;
    } else {
      prev->next = node;
    }
    prev = node;
  }
  file->head = head;
  strcpy(file->name, "LinkedFile");
  printf("Linked file '%s' allocated with blocks: ", file->name);
  LinkedNode *current = head;
  while (current != NULL) {
    printf("%d ", current->block);
    current = current->next;
  }
  printf("\n");
}

int main() {
  initialize_disk();
  print_disk();

  SequentialFile seq_file;
  int start_block = 2;
  int size = 4;
  allocate_sequential(&seq_file, start_block, size);
  print_disk();

  IndexedFile idx_file;
  int index_blocks[] = {6, 7, 8};
  allocate_indexed(&idx_file, index_blocks, 3);
  print_disk();

  LinkedFile lnk_file;
  int linked_blocks[] = {1, 3, 5};
  allocate_linked(&lnk_file, linked_blocks, 3);
  print_disk();

  return 0;
}
```