| #include <stdio.h>  #include <stdlib.h>  int main() {  int \*arr;  int n = 5;  // Dynamically allocate memory for an array of integers  arr = (int \*)malloc(n \* sizeof(int));  if (arr == NULL) {  printf("Memory allocation failed\n");  return 1;  }  // Initialize and display the array  for (int i = 0; i < n; i++) {  arr[i] = i + 1;  printf("arr[%d] = %d\n", i, arr[i]);  }  // Free the allocated memory  free(arr);  return 0;  } |
| --- |

| #include <stdio.h>  #include <unistd.h>  #include <sys/types.h>  int main() {  pid\_t pid;  // Create a new process  pid = fork();  if (pid < 0) {  // Fork failed  perror("Fork failed");  return 1;  } else if (pid == 0) {  // Child process  execlp("ls", "ls", "-l", NULL); // Execute 'ls -l'  perror("exec failed"); // exec only returns on failure  return 1;  } else {  // Parent process  printf("Parent process: Child PID is %d\n", pid);  }  return 0;  } |
| --- |

#### 

| #include <stdio.h>  #include <stdlib.h>  #include <fcntl.h>  #include <sys/mman.h>  #include <unistd.h>  #include <string.h>  int main() {  const char \*filepath = "example.txt";  int fd;  char \*mapped;  // Open the file  fd = open(filepath, O\_RDWR);  if (fd == -1) {  perror("Failed to open file");  return 1;  }  // Map the file into memory  mapped = mmap(NULL, 100, PROT\_READ | PROT\_WRITE, MAP\_SHARED, fd, 0);  if (mapped == MAP\_FAILED) {  perror("Failed to map file");  close(fd);  return 1;  }  // Write to the mapped memory  strcpy(mapped, "Hello, mmap!");  // Print the content of the mapped memory  printf("Mapped content: %s\n", mapped);  // Unmap and close the file  munmap(mapped, 100);  close(fd);  return 0;  } |
| --- |