C Programming in Linux Tutorial #036 – Shared Memory

En este tutorial, exploraremos cómo utilizar la memoria compartida en C en el entorno Linux. La memoria compartida es una técnica de IPC (comunicación entre procesos) que permite que varios procesos accedan a una región común de memoria.

¿Qué es la memoria compartida?

La memoria compartida es un mecanismo que permite que varios procesos accedan a la misma región de memoria en un espacio de direcciones compartido. Es una forma eficiente de compartir datos entre procesos porque evita la necesidad de copiar datos entre procesos.

Uso de memoria compartida en Linux

Para trabajar con memoria compartida en Linux, utilizamos las funciones proporcionadas por la biblioteca estándar de POSIX, como shm_open, ftruncate, mmap, y munmap. A continuación, se detalla el proceso.

1. Crear y abrir un objeto de memoria compartida

Primero, necesitamos crear o abrir un objeto de memoria compartida utilizando shm_open.

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <sys/mman.h>
#include <sys/stat.h&gt>
#include <unistd.h>
```

```
int main() {
    const char *name = "/my_shared_memory";
    const size_t SIZE = 4096;
    // Crear y abrir un objeto de memoria compartida
    int shm_fd = shm_open(name, O_CREAT | O_RDWR, 0666);
    if (shm_fd == -1) {
        perror("shm_open");
        exit(1);
    }
    // Ajustar el tamaño del objeto de memoria compartida
    if (ftruncate(shm_fd, SIZE) == -1) {
        perror("ftruncate");
        exit(1);
    }

    // Mapear el objeto de memoria compartida en el espacio de
direcciones del proceso
     void *ptr = mmap(0, SIZE, PROT_READ | PROT_WRITE,
MAP_SHARED, shm_fd, 0);
    if (ptr == MAP_FAILED) {
        perror("mmap");
        exit(1);
    }

    // Escribir en la memoria compartida
    sprintf((char *)ptr, "Hola desde la memoria compartida!");

    // Unmap y cerrar
    if (munmap(ptr, SIZE) == -1) {
        perror("munmap");
        exit(1);
    }
    close(shm_fd);
    return 0;
}
```

2. Leer desde la memoria compartida

En un segundo proceso, puedes leer desde la misma región de memoria compartida.

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <sys/mman.h>
#include <sys/stat.h&gt>
#include <unistd.h>

int main() {
    const char *name = "/my_shared_memory";
    const size_t SIZE = 4096;

    // Abrir el objeto de memoria compartida
    int shm_fd = shm_open(name, O_RDONLY, 0666);
    if (shm_fd == -1) {
        perror("shm_open");
        exit(1);
    }

    // Mapear el objeto de memoria compartida en el espacio de
direcciones del proceso
    void *ptr = mmap(0, SIZE, PROT_READ, MAP_SHARED, shm_fd,
0);
    if (ptr == MAP_FAILED) {
        perror("mmap");
        exit(1);
    }

    // Leer desde la memoria compartida
    printf("Mensaje desde la memoria compartida: %s\n", (char
*)ptr);

    // Unmap y cerrar
    if (munmap(ptr, SIZE) == -1) {
        perror("munmap");
        exit(1);
    }

    close(shm_fd);
    return 0;
}
```

3. Eliminar el objeto de memoria compartida

Una vez que ya no necesitas el objeto de memoria compartida, puedes eliminarlo con shm_unlink.

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <sys/mman.h>
#include <sys/stat.h&gt>
#include <unistd.h>

int main() {
    const char *name = "/my_shared_memory";

    // Eliminar el objeto de memoria compartida
    if (shm_unlink(name) == -1) {
        perror("shm_unlink");
        exit(1);
    }

    return 0;
}
```

Consideraciones

La memoria compartida es eficiente pero debe ser utilizada con cuidado para evitar problemas de sincronización y coherencia. Además, asegúrate de liberar los recursos de memoria compartida cuando ya no los necesites.

Conclusión

La memoria compartida es una técnica potente para la comunicación entre procesos en Linux. Con los ejemplos proporcionados, ahora deberías tener una comprensión básica de cómo crear, leer y eliminar objetos de memoria compartida en

C.