# Trabajo Práctico 2

Instituto Tecnológico de Buenos Aires - Sistemas Operativos (72.11)

Grupo 19


| **Ignacio Searles** | **Augusto Barthelemy Solá** | **Santiago Bassi** |
|---|---|---|
| isearles@itba.edu.ar | abarthelemysola@itba.edu.ar | sabassi@itba.edu.ar |
| 64.536 | 64.502 | 64.643 |


23 de septiembre de 2024


**Resumen**

El presente informe trata sobre el desarrollo de un kernel que administra los recursos de hardware de una computadora y que tiene una API para interactuar con el espacio de usuario. En el espacio de usuario se desarrolló un shell que permite ejecutar diferentes módulos cuyo objetivo es mostrar el funcionamiento del sistema.


## 1 Memory Manager

Ambos memory managers, al momento de su inicialización, reciben la cantidad de memoria que van a administrar. De esa memoria, consumen una parte para almacenar los datos necesarios para su funcionamiento.

Para los testeos del memory manager, se decidió que cuando se compila fuera del kernel, este corra indefinidamente, ya que se puede terminar la ejecución matando el proceso desde la shell. En cambio, cuando se compilan y ejecutan los tests dentro del kernel, se decidió que corran una cantidad determinada de veces, dado que el sistema implementado no posee procesos y no es posible frenar la ejecución del test.

### 1.1 Bitmap Memory Manager

Para el bitmap, se decidió usar dos bits para representar cada bloque de memoria, lo que permite utilizar tres estados (FREE, USED, START). A cada bloque se le asignó un tamaño definido por la macro BLOCK_SIZE.

A continuación, se detallan las instrucciones para compilar y ejecutar los tests del bitmap memory manager:

1. Compilación y ejecución de los tests dentro del kernel:
   (a) En la raíz del proyecto, ejecutar **make**.
   (b) Luego, ejecutar **./run.sh**.
   (c) En la shell que se abre, ejecutar **test_mm**.
2. Compilación y ejecución de los tests fuera del kernel:
   (a) En la raíz del proyecto, ejecutar **make bitmaptest**.

(b) Luego, ejecutar **cd Testing**.

(c) Por último, ejecutar **./bitmapTest <memoryAmount>**.
Siendo memoryAmount la cantidad de memoria que se desea asignar.

## 1.2 Buddy Memory Manager

Para el buddy, se decidió almacenar la información en un árbol, donde cada nodo almacena uno de los siguientes estados: FREE, SPLIT o USED. Al no tener memoria dinámica para generar el árbol, se decidió almacenarlo en un bloque de memoria contigua, disponiendo sus elementos en orden preorder.

A continuación, se detallan las instrucciones para compilar y ejecutar los tests del buddy memory manager:

1. Compilación y ejecución de los tests dentro del kernel:

   (a) En la raíz del proyecto, ejecutar **make buddy**.

   (b) Luego, ejecutar **./run.sh**.

   (c) En la shell que se abre, ejecutar **test_mm**.

2. Compilación y ejecución de los tests fuera del kernel:

   (a) En la raíz del proyecto, ejecutar **make buddytest**.

   (b) Luego, ejecutar **cd Testing**.

   (c) Por último, ejecutar **./buddytest <memoryAmount>**.
   Siendo memoryAmount la cantidad de memoria que se desea asignar.