OS'25 Project

PART V: **TESTING I**

(GROUP MODULES)

Agenda

- **CODE UPDATE**
- **PART V: TESTING OF GROUP MODULES**
  - Dynamic Allocator
  - Kernel Heap
  - Fault Handler I

CAUTION

During your solution, any SHARED data MUST be PROTECTED by critical section via LOCKS

ACTION

CORRECTNESS by DESIGN

Be LOGIC-DRIVEN... Not TEST-DRIVEN

Code Updates

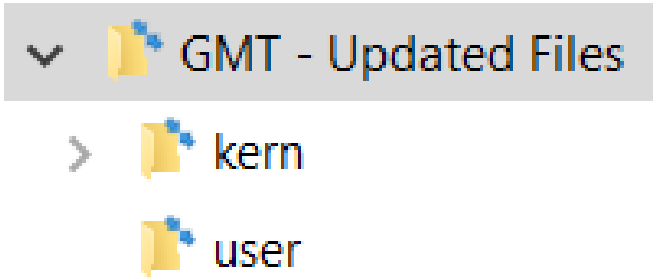
GROUP MODULES TEST

New Files

1. **SELECT ALL** in the given "**GMT - Updated Files**" folder,

2. **COPY** & **PASTE** (**REPLACE ALL**) in **FOS_CODES/FOS_PROJECT_2025_TEMPLATE/**

NOTE: If any of these files are already edited by you, make sure to apply the edits in the new files



Code Fix

In **`page_fault_handler()`** function inside **`kern/trap/fault_handler.c`** apply the following fix:

```
else if (isPageReplacmentAlgorithmOPTIMAL())
{
    //TODO: [PROJECT'25.IM#1] FAULT HANDLER II - #3 Clock Replacement
    //Your code is here
    //Comment the following line
    panic("page_fault_handler().REPLACEMENT is not implemented yet...!!");
}
```

```
else if (isPageReplacmentAlgorithmCLOCK())
{
    //TODO: [PROJECT'25.IM#1] FAULT HANDLER II - #3 Clock Replacement
    //Your code is here
    //Comment the following line
    panic("page_fault_handler().REPLACEMENT is not implemented yet...!!");
}
```

TEST: Dynamic Allocator

GROUP MODULES TEST

TEST: Dynamic Allocator

IMPORTANT NOTE

Before testing this module, you MUST DO **disable** the kernel heap:

- In '**inc/memlayout.h**': set **`USE_KHEAP`** to **0**

TEST: Dynamic Allocator

Module	Function	Diff.	Testing	Notes	Test Files
Dynamic Allocator	initialize	L1	`FOS> tst dynalloc init`	Fresh run	kern/tests/ test_dynamic_allocator.c
	alloc_block	L2	`FOS> tst dynalloc alloc`	Fresh run	
	get_block_size free_block	L1 L3	`FOS> tst dynalloc free`	Fresh run Depend on alloc_block	

`"test [TEST NAME] completed. Evaluation = …%"`

To ensure the test success, a congratulations message like this **MUST appear without any ERROR messages or PANICs**.

TEST: Kernel Heap

GROUP MODULES TEST

TEST: Kernel Heap

IMPORTANT NOTE

Before testing this module, you MUST DO **enable** the kernel heap:

- In '**inc/memlayout.h**': set **`USE_KHEAP`** to **1**

TEST: Kernel Heap

Module	Function	Diff.	Testing	Notes	Test Files
Kernel Heap	Kmalloc: Page Allocator	L3	`FOS> tst kheap cf kmalloc page`	Fresh run	kern/tests/ test_kheap.c
	Kmalloc: Block Allocator	L1	`FOS> tst kheap cf kmalloc blk`	Fresh run Depend on Dyn Alloc	
	Kfree: Page Allocator	L2	`FOS> tst kheap cf kfree page`	Fresh run Depend on kmalloc Test Custom Fit Strategy	
	Kfree: Block Allocator	L1	`FOS> tst kheap cf kfree blk`	Fresh run Depend on Dyn Alloc	
	kheap_virtual_address	L2	`FOS> tst kheap cf kvirtaddr`	Fresh run Depend on kmalloc & kfree	
	kheap_physical_address	L1	`FOS> tst kheap cf kphysaddr`		

`"test [TEST NAME] completed. Evaluation = …%"`

To ensure the test success, a congratulations message like this **MUST appear without any ERROR messages or PANICs**.

TEST: Fault Handler I

GROUP MODULES TEST

TEST: Fault Handler I

IMPORTANT NOTES

1. **Testing** depends on the KERNEL HEAP
2. **MUST enable** the kernel heap:

- In '**inc/memlayout.h**': set **`USE_KHEAP`** to **1**

TEST: Fault Handler I

Module	Function	Diff.	Testing	Notes	Test Files
Fault Handler I (Place)	Kernel Dyn. Alloc for a Process	L1	Already tested in Placement test	Depend on kmalloc	user/ tst_invalid_access.c
	Check Invalid Pointers	L1	`FOS> run tia 15`	tests invalid pointers	tst_invalid_access_1.c tst_invalid_access_2.c
	Fault handler (Placement + User Kern Stack)	L2	`FOS> run tpp 22`	tests page faults on stack + placement	tst_invalid_access_3.c tst_invalid_access_4.c tst_placement.c

"test [TEST NAME] completed. Evaluation = …%"

To ensure the test success, a congratulations message like this **MUST appear without any ERROR messages or PANICs**.

GOOD LUCK isA

☺ Enjoy **developing** your **own OS** ☺