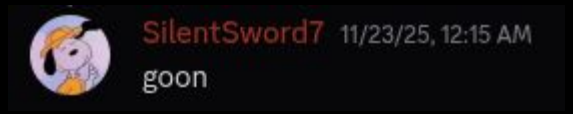
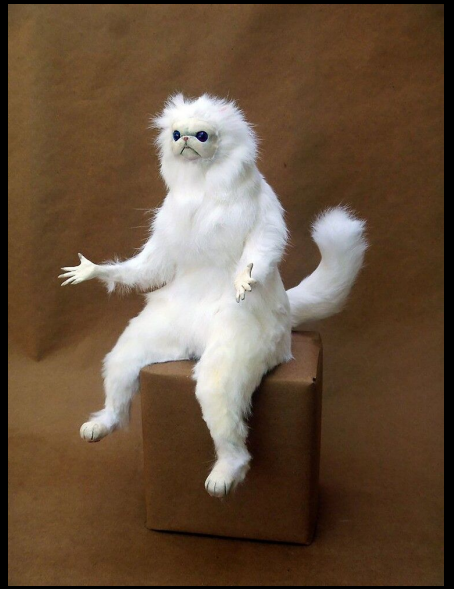
A Pretty Good File System and Coalescer

TEAM GOONIX

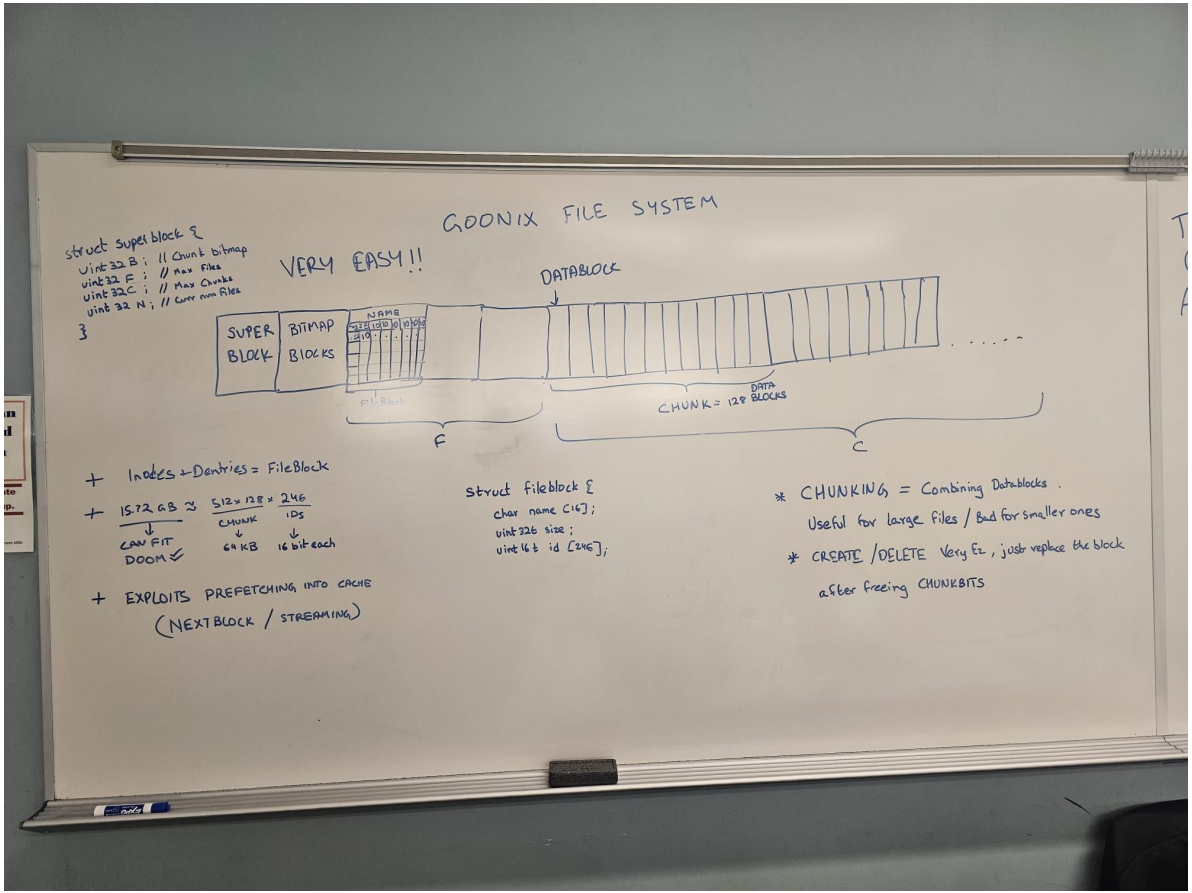


01

why?



The Initial Brainchild



Features

01 SIMPLICITY

A file system (in our opinion) is supposed to be a utility that doesn't take up the entire mp

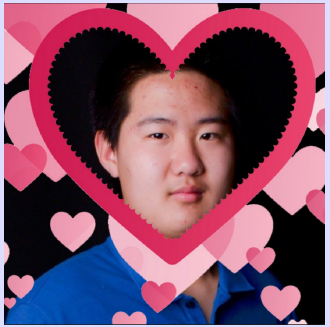
02 MICRO-OPTS

Could potentially we very good when implementing a prefetcher

03 IDK SEEMED COOL

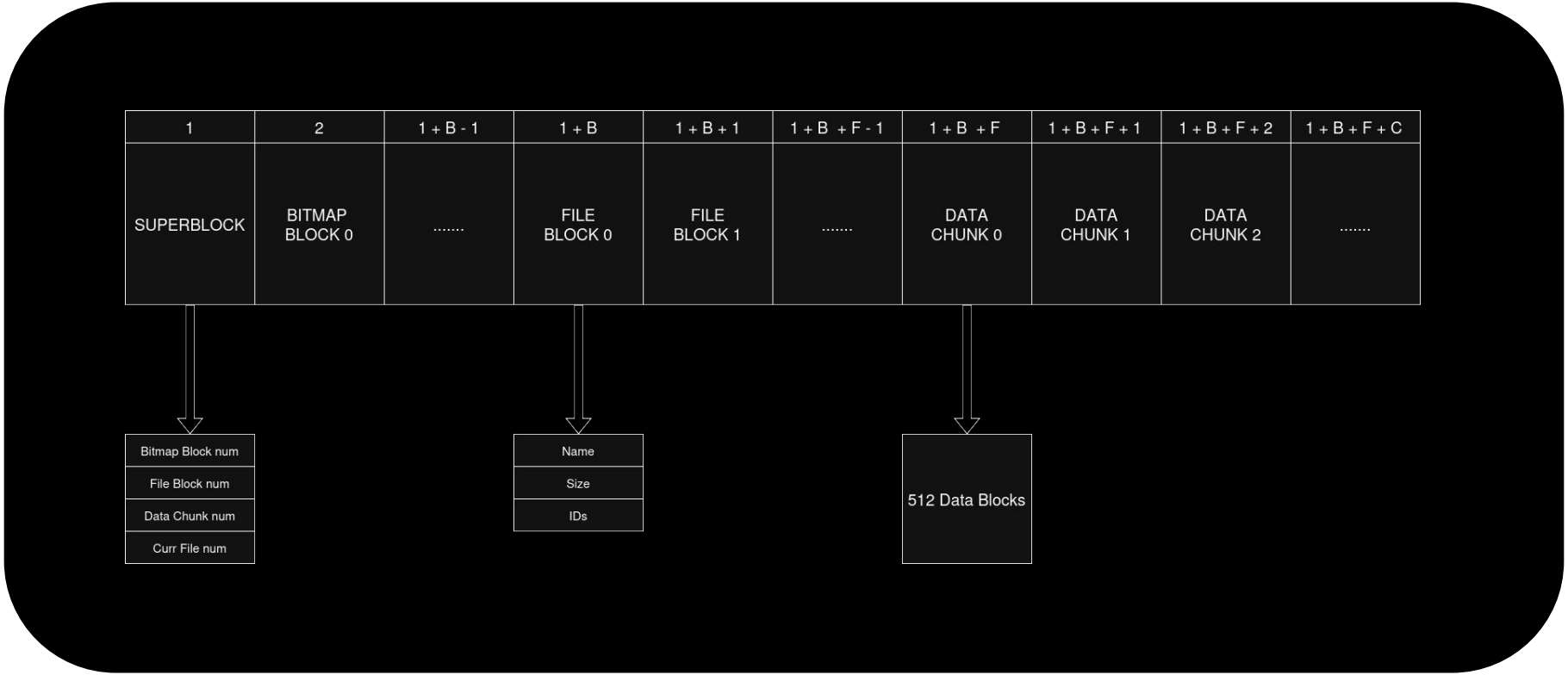
BASH Scripting my beloved

PLS NOTE THIS IS NOT COMBATIVE AND I LIKE KTFS BUT I WOULD LIKE TO MAKE IT EASIER SO THAT STUDENTS CAN WORK ON "COOLER/MORE IMPORTANT" PARTS OF THE OS.
Keegan is Chill



michael 12/9/25, 3:48 PM
no indirector or doubly indirect

Ninjapod 12/9/25, 3:49 PM
how big does an inode need to be to fit 16MB with only direct blocks



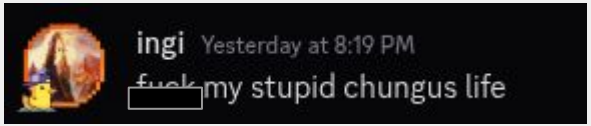
inodesb4hoez Yesterday at 2:37 PM

u get gfs actually working i'll put goonix sticker on my laptop bet

02

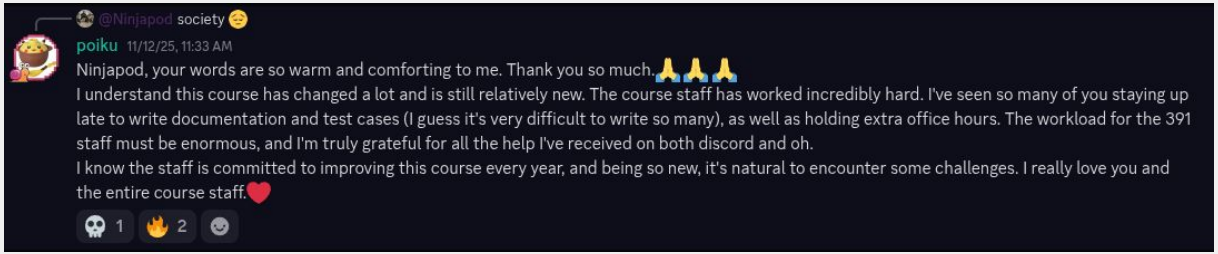
how?

MKFS_PGFS.sh

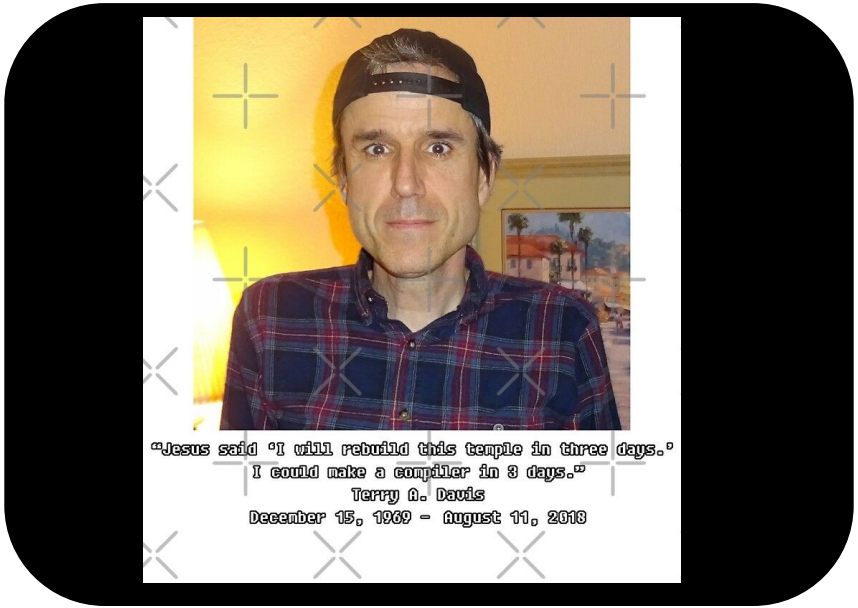


Just a quick BASH script I whipped up to make the file system image. I was severely skill issued and didn't understand bash well, so this took me a good amount of time

ktfs.c /h



I I would like to Preface that I am a very slow code writer (a pathetic 45 wpm) and I rewrote the entirety of ktfs.c /h like 5 hours ago and surely this means that given 1 checkpoint people can finish it.



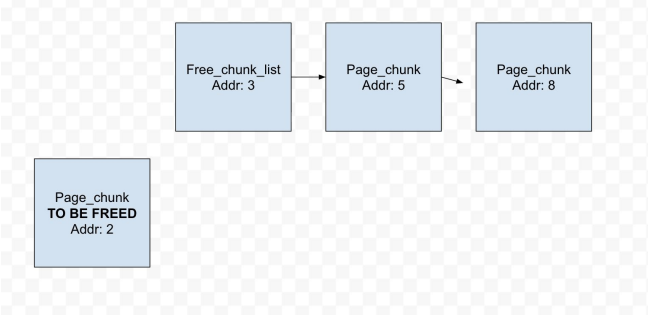
03

Memory coalescer (goon_gluer 9000)

memory.c /h

Free list orders freed memory from least to greatest.

Example

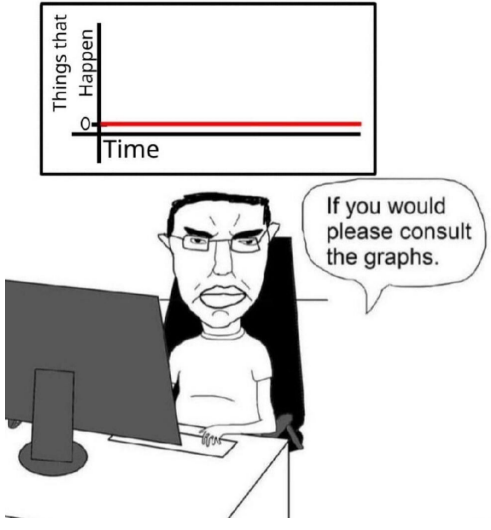



```
    if (free_chunk_list == NULL) {

        struct page_chunk *node = (struct page_chunk*)pp;
        node->pagecnt = 1;
        node->next = NULL;
        free_chunk_list = node;
        sfence_vma();
    return;
}
    while(temp_check != NULL) {
        if(((pp >= (void*)temp_prev) && (pp < temp))) {
            if(temp_prev == NULL) {
                goto head_insertion;
            }
            struct page_chunk* pp_chunk = (struct page_chunk*)pp;
            pp_chunk->pagecnt = 1;
            temp_prev->next = pp_chunk;
            pp_chunk->next = temp_check;
            sfence_vma();
            return;

        } else {
            head_insertion:
            if(temp_prev == NULL && (pp < temp)) {
                //insertion at the head
                struct page_chunk* pp_chunk = (struct page_chunk*)pp;
                pp_chunk->pagecnt = 1;
                pp_chunk->next = temp_check;
                free_chunk_list = pp_chunk;
                sfence_vma();
                return;

            }
        }
        //otherwise keep moving

         temp_check = (struct page_chunk*)temp;
         temp_prev = temp_check;
         temp_check = temp_check->next;
         temp = (void*)temp_check;
```

memory.c /h

```
//returns 0 if ordered from least to greatest

//returns -1 if not

//its like 5 am dude wtf am I doing with my life
int ordered_correct(void) {
    struct page_chunk* temp = (struct page_chunk*)free_chunk_list;
    while(temp->next != NULL) {
        if(!((uintptr_t)temp->next > (uintptr_t)temp)) {
            return -1;;
        }
        temp = temp->next;
}
return 0;
}
```

So how did I test the free logic?

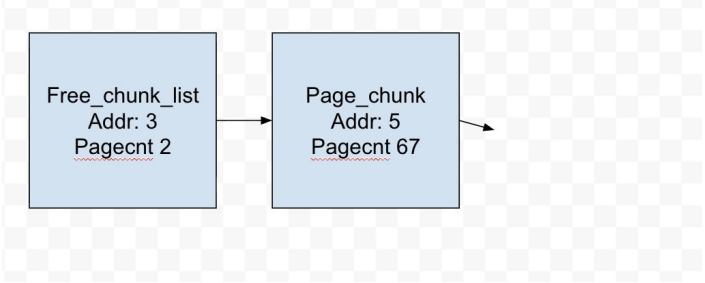



```
temp_pp1 = alloc_phys_page();
temp_pp2 = alloc_phys_pages(3);
temp_pp3 = alloc_phys_page();
void* temp_pp4 = alloc_phys_pages(69);
void* temp_pp5 = alloc_phys_pages(67);
void* temp_pp6 = alloc_phys_pages(2);

free_phys_pages(temp_pp4, 69);
free_phys_page(temp_pp1);
free_phys_page(temp_pp2);
free_phys_page(temp_pp3);
free_phys_pages(temp_pp5, 67);
free_phys_pages(temp_pp6, 2);
if(ordered_correct() == 0) {
 kprintf("FREE ABLE TO PLACE IN ORDERED MANNER! \n");
} else {
 kprintf("GOOOOOOOOOOON \n");
}
```

```
PREFORMING GOON GLUER TESTS!
RESET ACTIVE MSPACE WORKS!, GOT 1536
CURRENT HAVE 1531 PAGE LEFT
CURRENT HAVE 1536 PAGE LEFT
GOON GLUE BASIC FUNCTIONALITY WORKS!
FREE ABLE TO PLACE IN ORDERED MANNER!
```

memory.c /h

So how does this work?



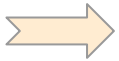

```c
int goon_gluer_9000(void) {
    if(free_chunk_list == NULL) {
        return GOON_NONEED_DEFRAG;
    }
    //check if everything is in the correct order
    struct page_chunk* temp = (struct page_chunk*)free_chunk_list;
    int big_goon = TRUE; //big goon is just telling us wether or not we have a big chunk
    while(temp->next != NULL) {
        if(!((uintptr_t)temp->next > (uintptr_t)temp)) {
            return GOON_GLUE_ERROR;
        }

        if(!((uintptr_t)temp + temp->pagecnt*PAGE_SIZE == (uintptr_t)temp->next)) {
            big_goon = FALSE;
        }
        temp = temp->next;
    }
    //verfied that everything is in the order that we want
    temp = (struct page_chunk*)free_chunk_list;

  int total_cnt =  free_phys_page_count();
  if(big_goon) {
    free_chunk_list->pagecnt = total_cnt;
    free_chunk_list->next = NULL;
    return 67;
  } //else we need to glue piece by piece
  while(temp->next != NULL) {
    struct page_chunk* comp = temp->next;

    if((uintptr_t)temp + temp->pagecnt * PAGE_SIZE == (uintptr_t) comp) {
        temp->pagecnt += comp->pagecnt;
        temp->next = comp->next;
    } else {
        temp = comp; //move tf on we aint coalescing shi for now
    }
  }


  return 67;
}
```

memory.c /h

```
int threadIdx = spawn_thread("goon1", goon_thread_func);
if (threadIdx < 0) {
    kprintf("spawn gooner 1failed: %d\n", threadIdx);
    return -69420;
}

int threadIdx2 = spawn_thread("worker2", goon_thread_func);
if (threadIdx2 < 0) {
    kprintf("spawn gooner2 failed: %d\n", threadIdx2);
    return -69420;
    kprintf("Jeśli to czytasz, to znaczy, że kocham Polskę. \n");
}

temp_pp1 = alloc_phys_page();
temp_pp2 = alloc_phys_pages(3);
temp_pp3 = alloc_phys_page();

free_phys_page(temp_pp1);
free_phys_pages(temp_pp2, 3);
free_phys_page(temp_pp3);
```

How did I test?

```
void* temp_pp1 = alloc_phys_page();
void* temp_pp2 = alloc_phys_pages(3);
void* temp_pp3 = alloc_phys_page();

kprintf("CURRENT HAVE %d PAGE LEFT \n", free_phys_page_count());

free_phys_page(temp_pp1);
free_phys_pages(temp_pp2, 3);
free_phys_page(temp_pp3);
kprintf("CURRENT HAVE %d PAGE LEFT \n", free_phys_page_count());
int checker = goon_gluer_9000();
if(checker != 67) {
    kprintf("ERROR IN GOON GLUE \n");
} else {
    if(free_phys_page_count() == ( 3 * MEGA_SIZE) / PAGE_SIZE) {
        kprintf("GOON GLUE BASIC FUNCTIONALITY WORKS! \n");
    } else {
        kprintf("GOON GLUE GOING WRONG, GETTING %d INSTEAD \n", free_phys_page_count());
    }
}
```



```
for (int i = 0; i < 10; ++i) {
    kprintf("main yielding for the (%d) time \n", i);
    running_thread_yield(); //some shit should happen ig
}

thread_join(threadIdx);
thread_join(threadIdx2);

if(perf_coa) {
    kprintf("SUCESSSSSSS COALESCER WORKS \n");
} else {
    kprintf("GOON GOON THREADING COALESCER DOESNT WORK  \n");
}
```

memory.c /h

How did I test?

```
int perf_coa(void) {
    if(free_chunk_list->next == NULL) {
        return 1;
    } else {
        return 0;
    }
}
```

```
FREE ABLE TO PLACE IN ORDERED MANNER!
main yielding for the (0) time
thread <goon1:1> starting
worker goon at loop 0 in (goon1:1)
thread <worker2:2> starting
worker goon at loop 0 in (worker2:2)
main yielding for the (1) time
worker goon at loop 1 in (goon1:1)
worker goon at loop 1 in (worker2:2)
main yielding for the (2) time
worker goon at loop 2 in (goon1:1)
worker goon at loop 2 in (worker2:2)
main yielding for the (3) time
worker goon at loop 3 in (goon1:1)
worker goon at loop 3 in (worker2:2)
main yielding for the (4) time
worker goon at loop 4 in (goon1:1)
worker goon at loop 4 in (worker2:2)
main yielding for the (5) time
gooner (goon1:1) has exited
gooner (worker2:2) has exited
main yielding for the (6) time
main yielding for the (7) time
main yielding for the (8) time
main yielding for the (9) time
SUCESSSSSSS COALESCER WORKS
```

Questions?


cyy 12/9/25, 3:57 PM
kevin fart when he quits course staff for the third time

THANK YOU

TEAM GOONIX