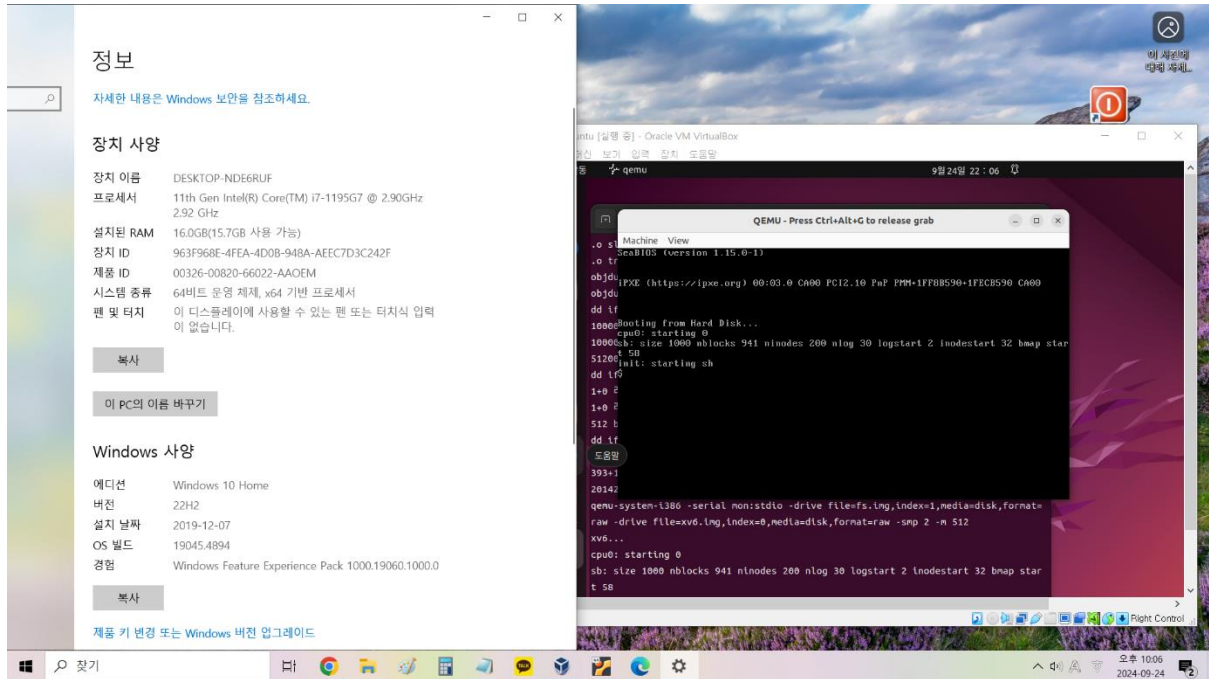


소속 : 소프트웨어학부

학번 : 20221993

이름 : 이재준

개발 환경



Lg gram 2022를 사용하고 있습니다. 사양은 위와 같습니다. 운영체제는 virtual machine을 이용한 우분투 리눅스를 사용했습니다.

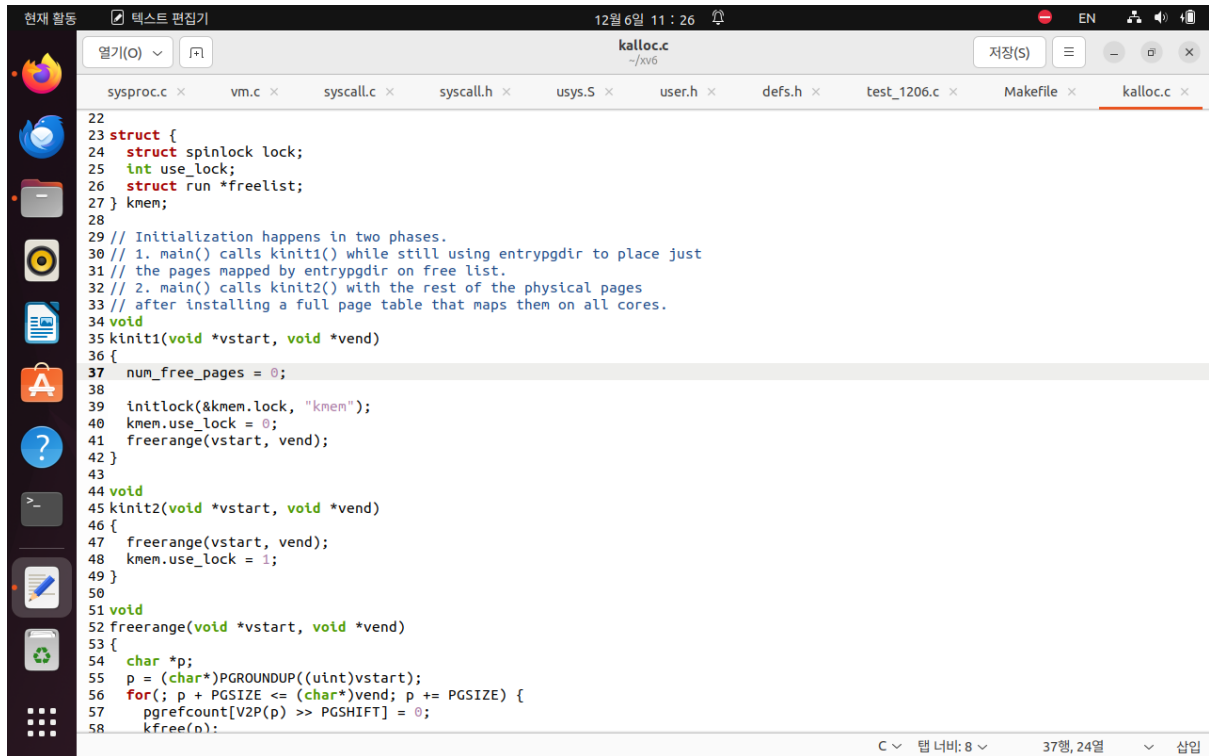
수정 및 작성한 코드에 대한 설명



```
1 // Physical memory allocator, intended to allocate
2 // memory for user processes, kernel stacks, page table pages,
3 // and pipe buffers. Allocates 4096-byte pages.
4
5 #include "types.h"
6 #include "defs.h"
7 #include "param.h"
8 #include "memlayout.h"
9 #include "mmu.h"
10 #include "spinlock.h"
11
12 unsigned int num_free_pages;
13 unsigned int pgrefcount[PHYSTOP >> PTXSHIFT];
14
15 void freerange(void *vstart, void *vend);
16 extern char end[]; // first address after kernel loaded from ELF file
17 // defined by the kernel linker script in kernel.ld
18
19 struct run {
20     struct run *next;
21 };
22
23 struct {
24     struct spinlock lock;
25     int use_lock;
26     struct run *freelist;
27 } kmem;
28
29 // Initialization happens in two phases.
30 // 1. main() calls kinit1() while still using entrypgdir to place just
31 // the pages mapped by entrypgdir on free list.
32 // 2. main() calls kinit2() with the rest of the physical pages
33 // after installing a full page table that maps them on all cores.
34 void
35 kinit1(void *vstart, void *vend)
36 {
37     num_free_pages = 0;
```

kalloc.c 에 전역 변수로 unsigned int num_free_pages 와 unsigned int pgrefcount[PHYSTOP >> PTXSHIFT]를 선언했습니다.

num_free_pages 는 여유 페이지를 나타내고 pgrefcount 는 물리 주소의 참조 값을 의미합니다.



```
22
23 struct {
24     struct spinlock lock;
25     int use_lock;
26     struct run *freelist;
27 } kmem;
28
29 // Initialization happens in two phases.
30 // 1. main() calls kinit1() while still using entrypgdir to place just
31 // the pages mapped by entrypgdir on free list.
32 // 2. main() calls kinit2() with the rest of the physical pages
33 // after installing a full page table that maps them on all cores.
34 void
35 kinit1(void *vstart, void *vend)
36 {
37     num_free_pages = 0;
38
39     initlock(&kmem.lock, "kmem");
40     kmem.use_lock = 0;
41     freerange(vstart, vend);
42 }
43
44 void
45 kinit2(void *vstart, void *vend)
46 {
47     freerange(vstart, vend);
48     kmem.use_lock = 1;
49 }
50
51 void
52 freerange(void *vstart, void *vend)
53 {
54     char *p;
55     p = (char*)PGROUNDUP((uint)vstart);
56     for(; p + PGSIZE <= (char*)vend; p += PGSIZE) {
57         pgfrecount[V2P(p) >> PGSIFT] = 0;
58         kfree(p);
59     }
60 }
```

kinit1()에서 num_free_pages 를 0 으로 초기화했습니다. kinit1()은 xv6 의 main()에서 초기에 호출되는 함수입니다.



```
64 // call to kalloc(). (The exception is when
65 // initializing the allocator; see kinit above.)
66 void
67 kfree(char *v)
68 {
69     struct run *r;
70
71     if((uint)v % PGSIZE || v < end || V2P(v) >= PHYSTOP)
72         panic("kfree");
73
74     // Fill with junk to catch dangling refs.
75     //!! memset(v, 1, PGSIZE);
76
77     if(kmem.use_lock)
78         acquire(&kmem.lock);
79     // r = (struct run *)v;
80     // r->next = kmem.freelist;
81     // kmem.freelist = r;
82
83     //!!
84     if (get_refcount(V2P(v)) > 0) {
85         dec_refcount(V2P(v));
86     }
87
88     if (get_refcount(V2P(v)) == 0) {
89         memset(v, 1, PGSIZE);
90         num_free_pages++;
91         r = (struct run *)v;
92         r->next = kmem.freelist;
93         kmem.freelist = r;
94     }
95
96     if(kmem.use_lock)
97         release(&kmem.lock);
98
99     // num_free_pages++;
100 }
```

kfree()에서 해당 페이지의 pgrefcount 에 +1 을 합니다. 그리고 pgrefcount 가 0 일
경우에만 freelist 를 추가시킬 수 있도록 합니다. 이 때 여유 페이지 수가 증가했으므로
numfreepages 에 +1 을 합니다.



```
194 }
195
196 if(kmem.use_lock)
197     release(&kmem.lock);
198 // num_free_pages++;
199 }
200 }
201
202 // Allocate one 4096-byte page of physical memory.
203 // Returns a pointer that the kernel can use.
204 // Returns 0 if the memory cannot be allocated.
205 char*
206 kalloc(void)
207 {
208     struct run *r;
209
210     if(kmem.use_lock)
211         acquire(&kmem.lock);
212
213     num_free_pages--;
214     r = kmem.freelist;
215
216     if(r) {
217         kmem.freelist = r->next;
218         pgrefcount[V2P((char *)r) >> PGSHIFT] = 1;
219     }
220     if(kmem.use_lock)
221         release(&kmem.lock);
222     return (char*)r;
223 }
224
225
226 unsigned int
227 get_refcount(unsigned int pa) {
228     return pgrefcount[pa >> PGSHIFT];
229 }
```

kalloc()에서 새로운 페이지를 할당할 때 num_free_pages 를 1 감소시킵니다. 그리고
해당 페이지에 대한 pgrefcount 는 1 로 초기화합니다.



```
107 {
108     struct run *r;
109
110     if(kmem.use_lock)
111         acquire(&kmem.lock);
112
113     num_free_pages--;
114     r = kmem.freelist;
115
116     if(r) {
117         kmem.freelist = r->next;
118         pgrefcount[V2P((char *)r) >> PGSHIFT] = 1;
119     }
120     if(kmem.use_lock)
121         release(&kmem.lock);
122
123     return (char*)r;
124 }
125
126 unsigned int
127 get_refcount(unsigned int pa) {
128     return pgrefcount[pa >> PGSHIFT];
129 }
130
131 void
132 inc_refcount(unsigned int pa) {
133     ++pgrefcount[pa >> PGSHIFT];
134
135     return ;
136 }
137
138 void
139 dec_refcount(unsigned int pa) {
140     --pgrefcount[pa >> PGSHIFT];
141
142     return ;
143 }
```

unsigned int get_refcount(unsigned int pa)와 void inc_refcount(unsigned int pa), void dec_refcount(unsigned int pa)를 구현했습니다. 각각 pgrefcount 를 반환하고, pgrefcount 에 +1 을 해주고 pgrefcount 에 -1 을 해줍니다.



```
313 // Given a parent process's page table, create a copy
314 // of it for a child.
315 pde_t*
316 copyvm(pde_t *pgdir, uint sz)
317 {
318     pde_t *d;
319     pte_t *pte;
320     uint pa, i, flags;
321     // char *mem;
322
323     if((d = setupkvm()) == 0)
324         return 0;
325     for(i = 0; i < sz; i += PGSIZE){
326         if(pte = walkpgdir(pgdir, (void *) i, 0)) == 0)
327             panic("copyvm: pte should exist");
328         if(!(*pte & PTE_P))
329             panic("copyvm: page not present");
330
331         *pte &= (~PTE_W);
332         pa = PTE_ADDR(*pte);
333         flags = PTE_FLAGS(*pte);
334         // if((mem = kalloc()) == 0)
335         //     goto bad;
336         // memmove(mem, (char*)P2V(pa), PGSIZE);
337         // if(mappages(d, (void*)i, PGSIZE, V2P(mem), flags) < 0) {
338         //     // kfree(mem);
339         //     goto bad;
340         // }
341
342         if (mappages(d, (void *)i, PGSIZE, pa, flags) < 0)
343             goto bad;
344
345         inc_refcount(pa);
346     }
347     lcr3(V2P(pgdir));
348     return d;
349 }
```

새로운 페이지를 meme 에 입력하는 과정과 할당된 페이지를 복사하는 과정을 생략했습니다. 그리고 Copy-on-Write 을 구현하기 위해 새로 매핑되는 자식 프로세스의 PTE 의 writeable flag 를 disable 시켰습니다. pgrefcount 에 +1 을 해주고 lcr3()를 통해 TLB 를 초기화시켰습니다.



```
32  ldt(ldt, sizeof(ltd));
33 }
34
35 //PAGEBREAK: 41
36 void
37 trap(struct trapframe *tf)
38 {
39     if(tf->trapno == T_SYSCALL){
40         if(myproc()->killed)
41             exit();
42         myproc()->tf = tf;
43         syscall();
44         if(myproc()->killed)
45             exit();
46         return;
47     }
48     switch(tf->trapno){
49     case T_PGFLT :
50         pagefault();
51         break;
52     case T_IRQ0 + IRQ_TIMER:
53         if(cpuid() == 0){
54             acquire(&tickslock);
55             ticks++;
56             wakeup(&ticks);
57             release(&tickslock);
58         }
59         lapiceoi();
60         break;
61     case T_IRQ0 + IRQ_IDE:
62         ideintr();
63         lapiceoi();
64         break;
65     case T_IRQ0 + IRQ_IDE+1:
66         // Bochs generates spurious IDE1 interrupts.
67         break;
68     }
```

pagefault 가 발생할 경우 pagefault()를 실행하도록 수정했습니다.



```
397 void
398 pagefault(void) {
399     unsigned int pagefault_VA;
400     pte_t *pte;
401     unsigned int rc, pa;
402
403     if ((pagefault_VA = rcr2()) < 0) {
404         panic("wrong approach");
405
406         return ;
407     }
408
409     pte = walkpgdir(myproc()->pgdir, (void *)pagefault_VA, 0);
410
411     pa = PTE_ADDR(*pte);
412     rc = get_refcount(pa);
413
414     if (rc > 1) {
415         char * mem;
416
417         if ((mem = kalloc()) == 0)
418             return ;
419
420         memmove(mem, (char *)P2V(pa), PGSIZE);
421         *pte = V2P(mem) | PTE_P | PTE_U | PTE_W;
422         dec_refcount(pa);
423     }
424     else if (rc == 1) {
425         *pte |= PTE_W;
426     }
427
428     lcr3(V2P(myproc()->pgdir));
429
430     return ;
431 }
432
```

pagefault 가 발생했을 때의 핸들러입니다. pagefault()는 pagefault 가 발생한 가상 주소를 가져와서 작동합니다. copyuvm()의 코드를 몇 줄 가져와 완성시켰습니다.



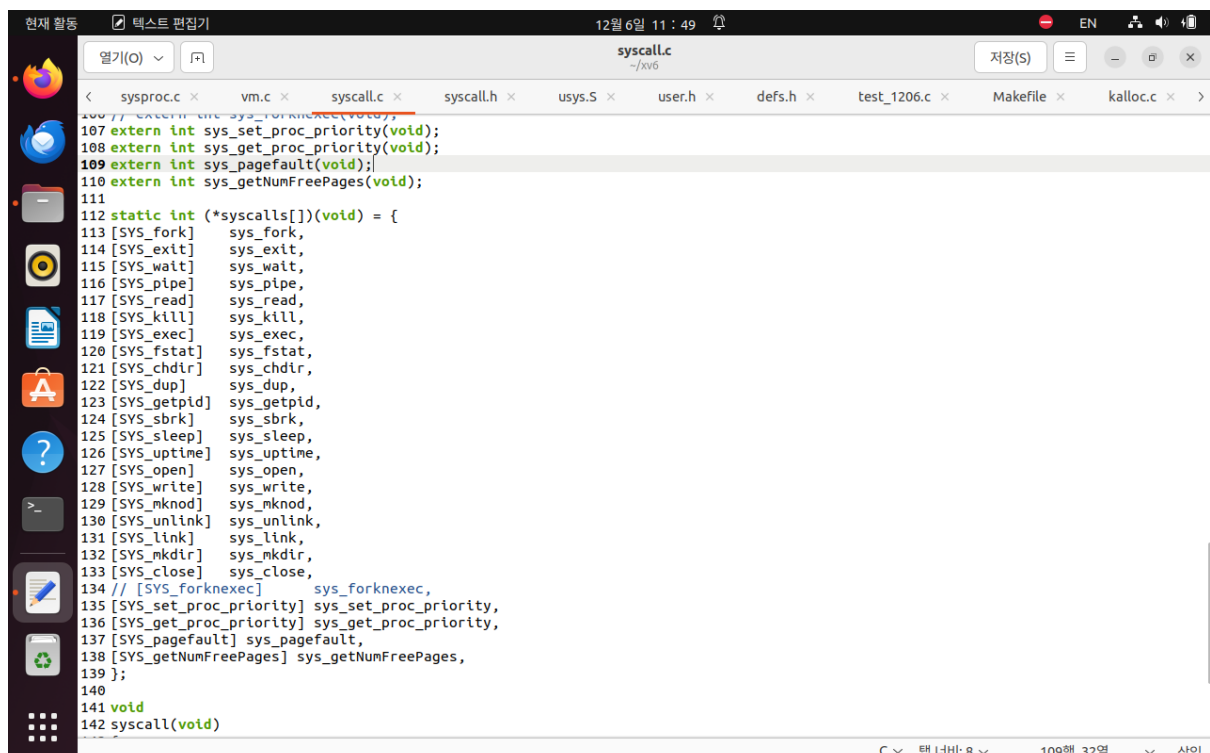
```
44 void
45 kinit2(void *vstart, void *vend)
46 {
47     freerange(vstart, vend);
48     kmem.use_lock = 1;
49 }
50
51 void
52 freerange(void *vstart, void *vend)
53 {
54     char *p;
55     p = (char*)PGROUNDUP((uint)vstart);
56     for(; p + PGSIZE <= (char*)vend; p += PGSIZE) {
57         pgrefcount[V2P(p) >> PGSHIFT] = 0;
58         kfree(p);
59     }
60 }
61 //PAGEBREAK: 21
62 // Free the page of physical memory pointed at by v,
63 // which normally should have been returned by a
64 // call to kalloc(). (The exception is when
65 // initializing the allocator; see kinit above.)
66 void
67 kfree(char *v)
68 {
69     struct run *r;
70
71     if((uint)v % PGSIZE || v < end || V2P(v) >= PHYSTOP)
72         panic("kfree");
73
74     // Fill with junk to catch dangling refs.
75     //!! memset(v, 1, PGSIZE);
76
77     if(kmem.use_lock)
78         acquire(&kmem.lock);
79     r = (struct run*)v;
80     r->next = kmem.freelist;
```


freerange()가 물리 페이지(프레임)를 초기화할 때 해당 페이지의 pgrefcount 도 초기화시켜줍니다.

이하는 pagefault 와 getNumFreePages 시스템 콜을 구현한 모습입니다.



```
132 if (argint(0, &id) < 0 || argint(1, &priority) < 0)
133     return -1;
134
135 return set_proc_priority(id, priority);
136 }
137
138 int
139 sys_get_proc_priority(void)
140 {
141     int id;
142
143     if (argint(0, &id) < 0)
144         return -1;
145
146     return get_proc_priority(id);
147 }
148
149 int
150 sys_pagefault(void)
151 {
152     unsigned int fault_addr;
153
154     if (argint(0, (int *)&fault_addr) < 0)
155         return -1;
156
157     pagefault();
158
159     return 0;
160 }
161
162 int
163 sys_getNumFreePages(void)
164 {
165     extern unsigned int num_free_pages;
166
167     return num_free_pages;
168 }
```



```
107 extern int sys_set_proc_priority(void);
108 extern int sys_get_proc_priority(void);
109 extern int sys_pagefault(void);
110 extern int sys_getNumFreePages(void);
111
112 static int (*syscalls[])(void) = {
113     [SYS_fork] sys_fork,
114     [SYS_exit] sys_exit,
115     [SYS_wait] sys_wait,
116     [SYS_plpe] sys_plpe,
117     [SYS_read] sys_read,
118     [SYS_kill] sys_kill,
119     [SYS_exec] sys_exec,
120     [SYS_fstat] sys_fstat,
121     [SYS_chdir] sys_chdir,
122     [SYS_dup] sys_dup,
123     [SYS_getpid] sys_getpid,
124     [SYS_sbrk] sys_sbrk,
125     [SYS_sleep] sys_sleep,
126     [SYS_uptime] sys_uptime,
127     [SYS_open] sys_open,
128     [SYS_write] sys_write,
129     [SYS_mknod] sys_mknod,
130     [SYS_unlink] sys_unlink,
131     [SYS_link] sys_link,
132     [SYS_mkdir] sys_mkdir,
133     [SYS_close] sys_close,
134     // [SYS_forknexec] sys_forknexec,
135     [SYS_set_proc_priority] sys_set_proc_priority,
136     [SYS_get_proc_priority] sys_get_proc_priority,
137     [SYS_pagefault] sys_pagefault,
138     [SYS_getNumFreePages] sys_getNumFreePages,
139 };
140
141 void
142 syscall(void)
```

현재 활동 텍스트 편집기 12월 6일 11:50 EN

syscall.h

열기(O) 🔍 저장(S) ⋮ ⌵ ⌶ ⌵

< sysproc.c x vm.c x syscall.c x syscall.h x usys.S x user.h x defs.h x test_1206.c x Makefile x kalloc.c x >

```
1 // System call numbers
2 #define SYS_fork 1
3 #define SYS_exit 2
4 #define SYS_wait 3
5 #define SYS_pipe 4
6 #define SYS_read 5
7 #define SYS_kill 6
8 #define SYS_exec 7
9 #define SYS_fstat 8
10 #define SYS_chdir 9
11 #define SYS_dup 10
12 #define SYS_getpid 11
13 #define SYS_sbrk 12
14 #define SYS_sleep 13
15 #define SYS_uptime 14
16 #define SYS_open 15
17 #define SYS_write 16
18 #define SYS_mknod 17
19 #define SYS_unlink 18
20 #define SYS_link 19
21 #define SYS_mkdir 20
22 #define SYS_close 21
23 // #define SYS_forknexec 22
24 #define SYS_set_proc_priority 23
25 #define SYS_get_proc_priority 24
26 #define SYS_getNumFreePages 25
27 #define SYS_pagefault 26
28
```

C/ObjC 헤더 ▾ 탭 너비: 8 ▾ 26행, 35열 ▾ 삼입

현재 활동 텍스트 편집기 12월 6일 11:50 EN

usys.S

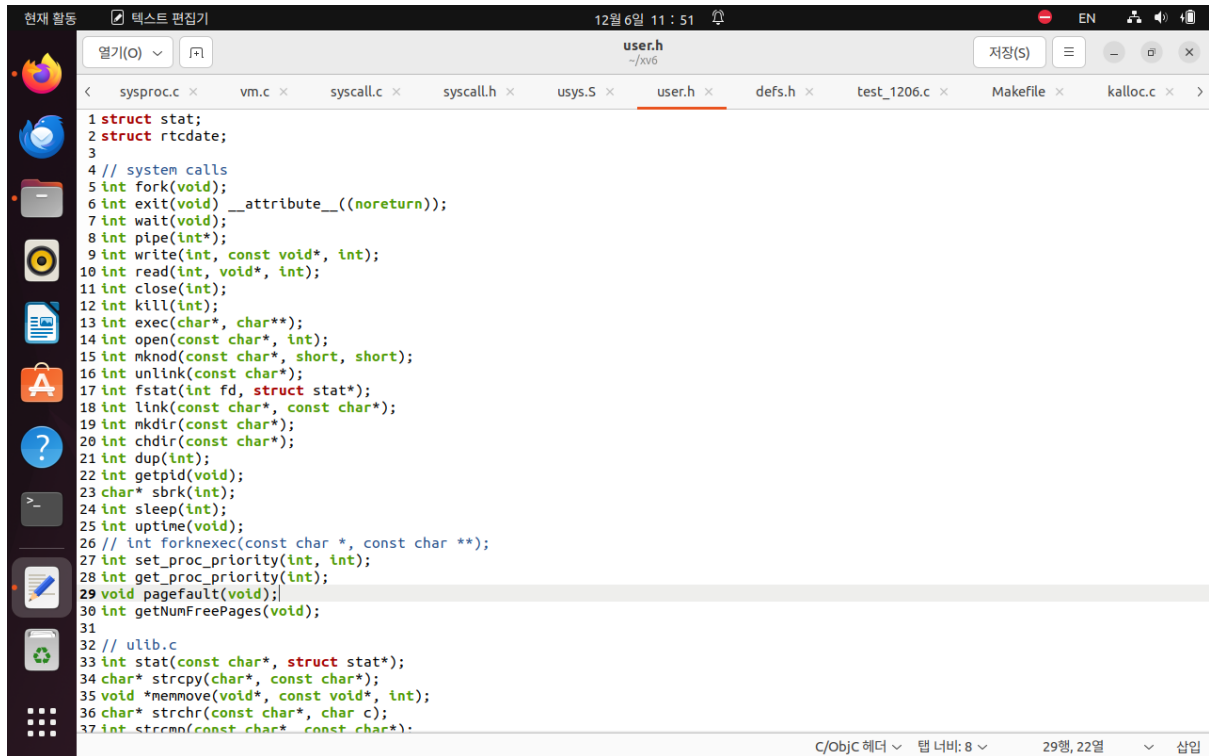
열기(O) 🔍 저장(S) ⋮ ⌵ ⌶ ⌵

< sysproc.c x vm.c x syscall.c x syscall.h x usys.S x user.h x defs.h x test_1206.c x Makefile x kalloc.c x >

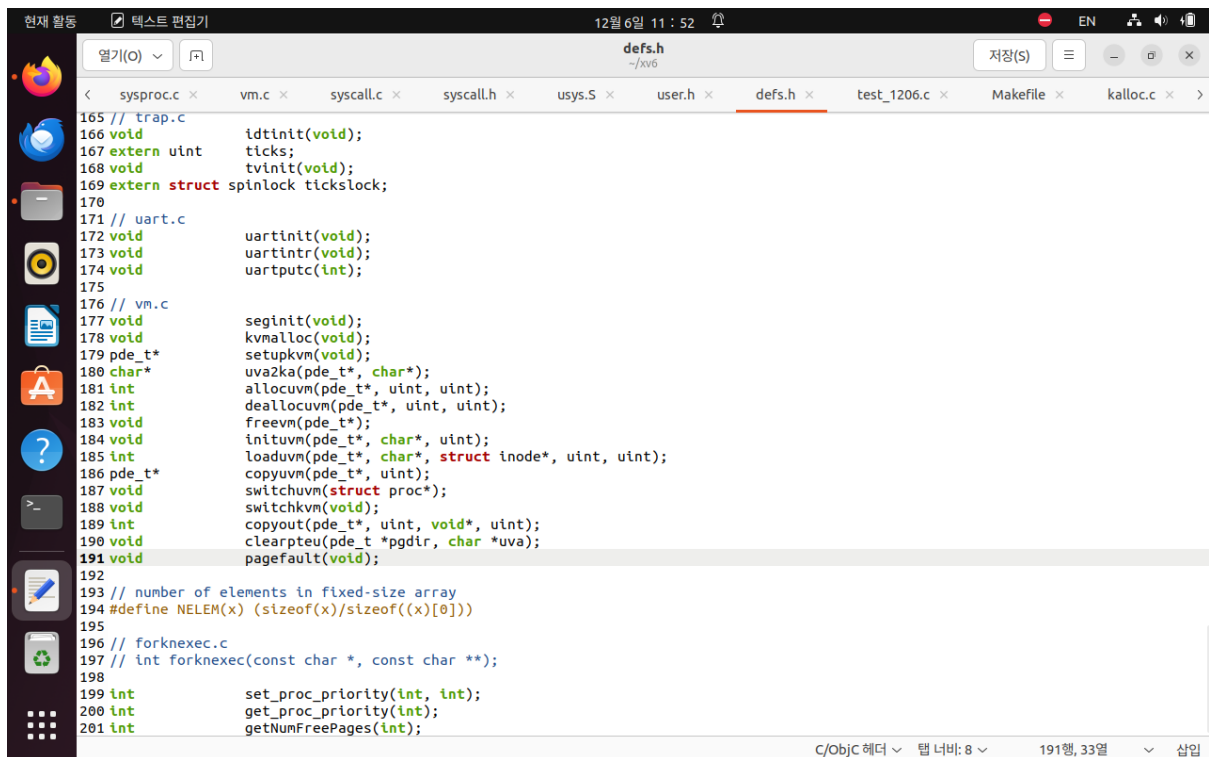
```
1 #include "syscall.h"
2 #include "traps.h"
3
4 #define SYSCALL(name) \
5 .globl name; \
6 name: \
7     movl $SYS_ ## name, %eax; \
8     int $T_SYSCALL; \
9     ret
10
11 SYSCALL(fork)
12 SYSCALL(exit)
13 SYSCALL(wait)
14 SYSCALL(pipe)
15 SYSCALL(read)
16 SYSCALL(write)
17 SYSCALL(close)
18 SYSCALL(kill)
19 SYSCALL(exec)
20 SYSCALL(open)
21 SYSCALL(mknod)
22 SYSCALL(unlink)
23 SYSCALL(fstat)
24 SYSCALL(link)
25 SYSCALL(mkdir)
26 SYSCALL(chdir)
27 SYSCALL(dup)
28 SYSCALL(getpid)
29 SYSCALL(sbrk)
30 SYSCALL(sleep)
31 SYSCALL(uptime)
32 // SYSCALL(forknexec)
33 SYSCALL(set_proc_priority)
34 SYSCALL(get_proc_priority)
35 SYSCALL(pagefault)
36 SYSCALL(getNumFreePages)
37
```

맞는 대괄호가 다음 줄에 있습니다: 35

C ▾ 탭 너비: 8 ▾ 35행, 19열 ▾ 삼입

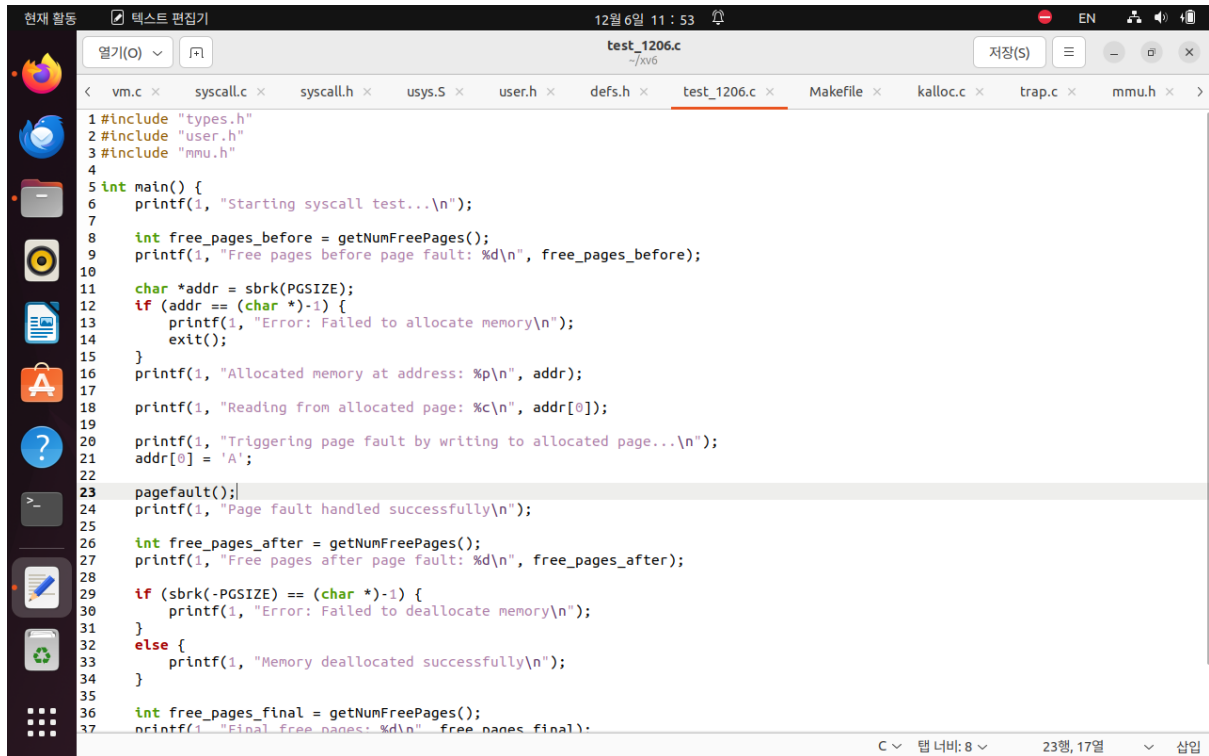


```
1 struct stat;
2 struct rtcdate;
3
4 // system calls
5 int fork(void);
6 int exit(void) __attribute__((noreturn));
7 int wait(void);
8 int pipe(int*);
9 int write(int, const void*, int);
10 int read(int, void*, int);
11 int close(int);
12 int kill(int);
13 int exec(char*, char**);
14 int open(const char*, int);
15 int mknod(const char*, short, short);
16 int unlink(const char*);
17 int fstat(int fd, struct stat*);
18 int link(const char*, const char*);
19 int mkdir(const char*);
20 int chdir(const char*);
21 int dup(int);
22 int getpid(void);
23 char* sbrk(int);
24 int sleep(int);
25 int uptime(void);
26 // int forknexec(const char *, const char **);
27 int set_proc_priority(int, int);
28 int get_proc_priority(int);
29 void pagefault(void);
30 int getNumFreePages(void);
31
32 // ulib.c
33 int stat(const char*, struct stat*);
34 char* strcpy(char*, const char*);
35 void *memmove(void*, const void*, int);
36 char* strchr(const char*, char c);
37 int strcmp(const char*, const char*);
```



```
165 // trap.c
166 void idtinit(void);
167 extern uint ticks;
168 void tvinit(void);
169 extern struct spinlock tickslock;
170
171 // uart.c
172 void uartinit(void);
173 void uartintr(void);
174 void uartputc(int);
175
176 // vm.c
177 void seginit(void);
178 void kvmalloc(void);
179 pde_t* setupkvm(void);
180 char* uva2ka(pde_t*, char*);
181 int allocuvm(pde_t*, uint, uint);
182 int deallocuvm(pde_t*, uint, uint);
183 void freevm(pde_t*);
184 void inituvm(pde_t*, char*, uint);
185 int loaduvm(pde_t*, char*, struct inode*, uint, uint);
186 pde_t* copyuvm(pde_t*, uint);
187 void switchuvm(struct proc*);
188 void switchkvm(void);
189 int copyout(pde_t*, uint, void*, uint);
190 void clearpteu(pde_t *pgdir, char *uva);
191 void pagefault(void);
192
193 // number of elements in fixed-size array
194 #define NELEM(x) (sizeof(x)/sizeof((x)[0]))
195
196 // forknexec.c
197 // int forknexec(const char *, const char **);
198
199 int set_proc_priority(int, int);
200 int get_proc_priority(int);
201 int getNumFreePages(int);
```

sysproc.c 에 구현한 핸들러를 실행시키는 코드를 작성했습니다. 그리고 syscall.c 에 해당 함수들을 전역 변수 선언했고 함수 포인터를 명시했습니다. syscall.h 에 시스템 콜의 번호를 매핑했고 usys.S 에 시스템 콜을 시스템 콜 목록에 추가했습니다. user.h 에 시스템 콜에 대한 정의를 했고 defs.h 에 시스템 콜이 다른 파일에서도 보이도록 했습니다.



```
1 #include "types.h"
2 #include "user.h"
3 #include "mmu.h"
4
5 int main() {
6     printf(1, "Starting syscall test...\n");
7
8     int free_pages_before = getNumFreePages();
9     printf(1, "Free pages before page fault: %d\n", free_pages_before);
10
11     char *addr = sbrk(PGSIZE);
12     if (addr == (char *)-1) {
13         printf(1, "Error: Failed to allocate memory\n");
14         exit();
15     }
16     printf(1, "Allocated memory at address: %p\n", addr);
17     printf(1, "Reading from allocated page: %c\n", addr[0]);
18
19     printf(1, "Triggering page fault by writing to allocated page...\n");
20     addr[0] = 'A';
21
22     pagefault();
23     printf(1, "Page fault handled successfully\n");
24
25     int free_pages_after = getNumFreePages();
26     printf(1, "Free pages after page fault: %d\n", free_pages_after);
27
28     if (sbrk(-PGSIZE) == (char *)-1) {
29         printf(1, "Error: Failed to deallocate memory\n");
30     }
31     else {
32         printf(1, "Memory deallocated successfully\n");
33     }
34
35     int free_pages_final = getNumFreePages();
36     printf(1, "Final free pages: %d\n", free_pages_final);
37 }
```

구현한 Copy - on - Write 을 확인하는 유저 프로그램입니다. 유저 프로그램은 여유로운 페이지 수를 시스템에서 반환 후 쓰기 작업으로 Copy - on - Write 을 처리하는 등을 수행합니다.

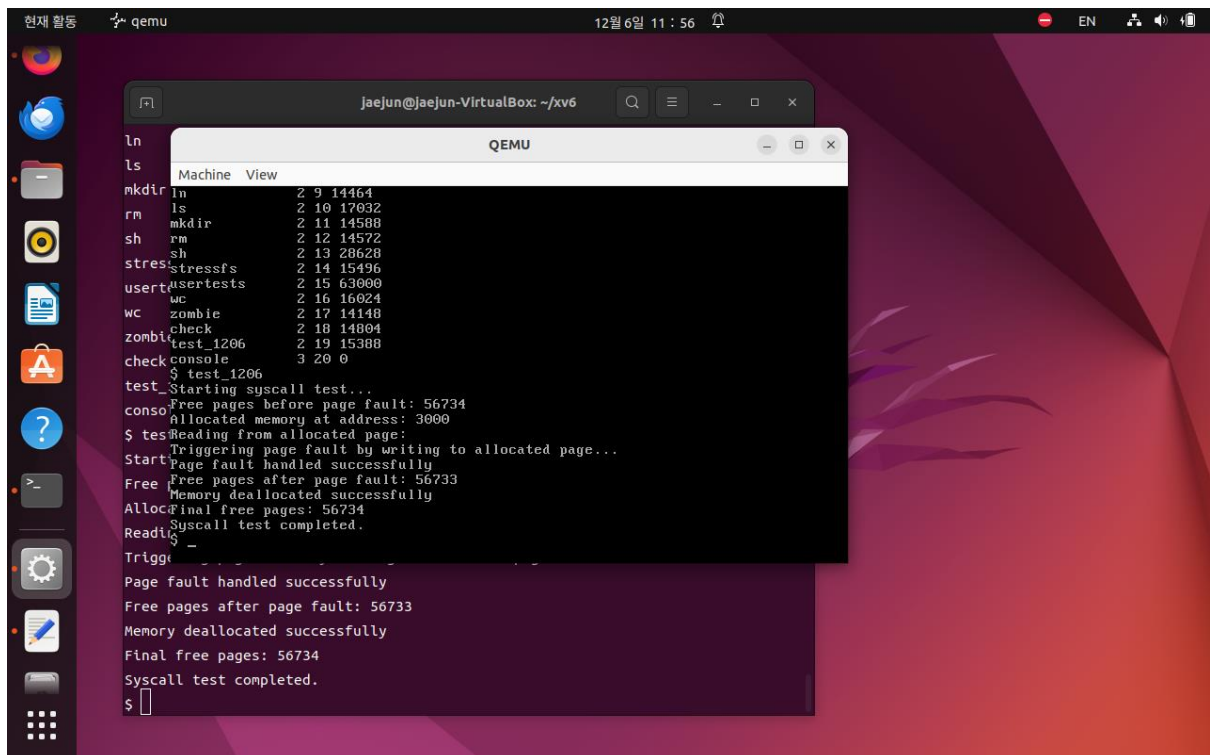


```
160 gcc -Werror -Wall -o mkfs mkfs.c
161
162 # Prevent deletion of intermediate files, e.g. cat.o, after first build, so
163 # that disk image changes after first build are persistent until clean. More
164 # details:
165 # http://www.gnu.org/software/make/manual/html_node/Chained-Rules.html
166 .PRECIOUS: %.o
167
168 UPROGS=\
169     _cat\
170     _echo\
171     _forktest\
172     _grep\
173     _init\
174     _kill\
175     _ln\
176     _ls\
177     _mkdir\
178     _rm\
179     _sh\
180     _stressfs\
181     _usertests\
182     _wc\
183     _zombie\
184     _check\
185     _test_1206\
186
187 fs.img: mkfs README $(UPROGS)
188     ./mkfs fs.img README $(UPROGS)
189
190 -include *.d
191
192 clean:
193     rm -f *.tex *.dvi *.idx *.aux *.log *.ind *.ilg \
194         *.o *.d *.asm *.sym vectors.S bootblock entryother \
195         initcode initcode.out kernel xv6.img fs.img kernelmemfs \
196         xv6memfs.img mkfs .ndhinit \
```



```
220 qemu: fs.img xv6.img
227 $(QEMU) -serial mon:stdio $(QEMUOPTS)
228
229 qemu-memfs: xv6memfs.img
230 $(QEMU) -drive file=xv6memfs.img,index=0,media=disk,format=raw -smp $(CPUS) -m 256
231
232 qemu-nox: fs.img xv6.img
233 $(QEMU) -nographic $(QEMUOPTS)
234
235 .gdbinit: .gdbinit.tmpl
236 sed "s/localhost:1234/localhost:$(GDBPORT)/" < $^ > $@
237
238 qemu-gdb: fs.img xv6.img .gdbinit
239 @echo "*** Now run 'gdb'." 1>&2
240 $(QEMU) -serial mon:stdio $(QEMUOPTS) -S $(QEMU gdb)
241
242 qemu-nox-gdb: fs.img xv6.img .gdbinit
243 @echo "*** Now run 'gdb'." 1>&2
244 $(QEMU) -nographic $(QEMUOPTS) -S $(QEMU gdb)
245
246 # CUT HERE
247 # prepare dist for students
248 # after running make dist, probably want to
249 # rename it to rev0 or rev1 or so on and then
250 # check in that version.
251
252 EXTRA=\
253     mkfs.c ulib.c user.h cat.c echo.c forktest.c grep.c kill.c \
254     ln.c ls.c mkdir.c rm.c stressfs.c usertests.c wc.c zombie.c \
255     printf.c umalloc.c \ check.c \ test_1206.c \
256     README dot-bochsrc *.pl toc.* runoff runoff1 runoff.list \
257     .gdbinit.tmpl gdbutil\
258
259 dist:
260     rm -rf dist
261     mkdir dist
262     for i in $(FILES); \
```

유저 프로그램을 컴파일하기 위한 MakeFile 입니다.



```
ln
ls
mkdir ln
ls
rm mkdir
sh rm
sh
stressfs
usertests
wc
wc zombie
zombie check
check test_1206
check console
$ test_1206
test_ Starting syscall test...
Free pages before page fault: 56734
conso Allocated memory at address: 3000
$ tes Reading from allocated page:
Start Triggering page fault by writing to allocated page...
Page fault handled successfully
Free Free pages after page fault: 56733
Memory deallocated successfully
Alloc Final free pages: 56734
Read Syscall test completed.
$ _
Trigg
Page fault handled successfully
Free pages after page fault: 56733
Memory deallocated successfully
Final free pages: 56734
Syscall test completed.
$
```

유저 프로그램의 실행 결과입니다.

과제 수행 중 발생한 문제점과 해결 방법

처음에는 기존에 있는 기호 상수를 사용하려고 했습니다. 그러나 나중에 소스 코드를 다시 볼 때 혼란이 있을 것을 염려하여 기호 상수를 새로 정의했습니다. mmu.h 에 #define PGSHIFT 를 선언했습니다. 페이지 이동을 위한 기호 상수입니다.