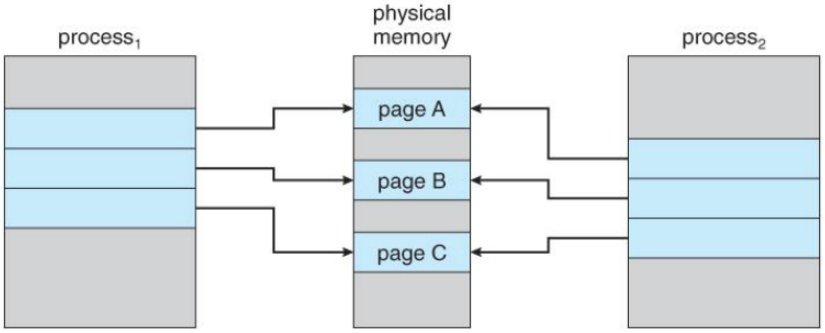
Project #4: Copy-On-Write

Instructor: Sungyong Ahn

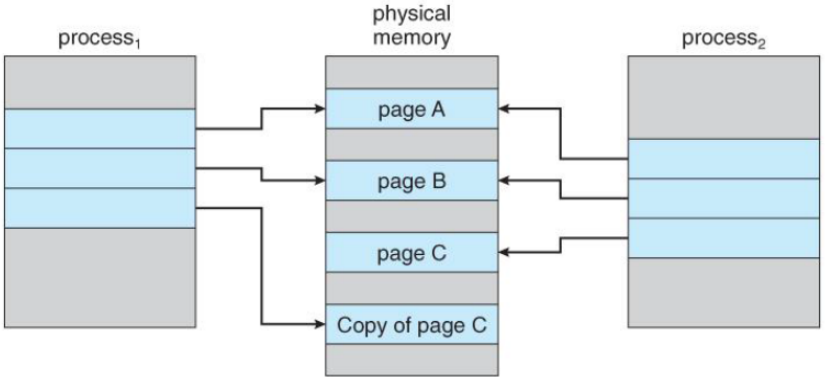
Copy-on-Write

- **When a process forks**
  - Create shared mappings to the same page frames in physical page
  - Shared pages are protected as read-only



Before process 1 modifies page C

- **When data is written to shared pages**
  - Protection fault is generated
  - OS allocates new space in physical memory and directs the write to it



After process 1 modifies page C

- **Reference counter for physical pages is needed**

Project #3. Copy-on-Write

- `proc.c`

```
int
fork(void)
{
  int i, pid;
  struct proc *np;
  struct proc *curproc = myproc();

  // Allocate process.
  if((np = allocproc()) == 0){
    return -1;
  }

  // Copy process state from proc.
  if((np->pgdir = copyuvm(curproc->pgdir, curproc->sz)) == 0){
    kfree(np->kstack);
    np->kstack = 0;
    np->state = UNUSED;
    return -1;
  }
```

vm.c
copyuvm()

```
pde_t*
copyuvm(pde_t *pgdir, uint sz)
{
  pde_t *d;
  pte_t *pte;
  uint pa, i, flags;
  char *mem;

  if((d = setupkvm()) == 0)
    return 0;
  for(i = 0; i < sz; i += PGSIZE){
    if((pte = walkpgdir(pgdir, (void *) i, 0)) == 0)
      panic("copyuvm: pte should exist");
    if(!(*pte & PTE_P))
      panic("copyuvm: page not present");
    pa = PTE_ADDR(*pte);
    flags = PTE_FLAGS(*pte);
    if((mem = kalloc()) == 0)
      goto bad;
    memmove(mem, (char*)P2V(pa), PGSIZE);
    if(mappages(d, (void*)i, PGSIZE, V2P(mem), flags) < 0)
      goto bad;
  }
  return d;

bad:
  freevm(d);
  return 0;
}
```

kernel address space를 위한 page table 생성

pde_t *pgdir은 parent process의 page table
pde_t *d는 새로운 child process를 위한 page table

vm.c
copyuvm()

```
pde_t*
copyuvm(pde_t *pgdir, uint sz)
{
  pde_t *d;
  pte_t *pte;
  uint pa, i, flags;
  char *mem;

  if((d = setupkvm()) == 0)
    return 0;
  for(i = 0; i < sz; i += PGSIZE){
    if((pte = walkpgdir(pgdir, (void *) i, 0)) == 0)
      panic("copyuvm: pte should exist");
    if(!(*pte & PTE_P))
      panic("copyuvm: page not present");
    pa = PTE_ADDR(*pte);
    flags = PTE_FLAGS(*pte);
    if((mem = kalloc()) == 0)
      goto bad;
    memmove(mem, (char*)P2V(pa), PGSIZE);
    if(mappages(d, (void*)i, PGSIZE, V2P(mem), flags)
      goto bad;
  }
  return d;

bad:
  freevm(d);
  return 0;
}
```

pde_t *pgdir은 parent process의 page table
pde_t *d는 새로운 child process를 위한 page table

virtual address에 대한 page table entry 반환

page table entry에서 user virtual address에 대한 physical address를 구함

주의: pa는 parent process의 physical page address

vm.c
copyuvm()

```
pde_t*
copyuvm(pde_t *pgdir, uint sz)
{
  pde_t *d;
  pte_t *pte;
  uint pa, i, flags;
  char *mem;

  if((d = setupkvm()) == 0)
    return 0;
  for(i = 0; i < sz; i += PGSIZE){
    if((pte = walkpgdir(pgdir, (void *) i, 0)) == 0)
      panic("copyuvm: pte should exist");
    if(!(*pte & PTE_P))
      panic("copyuvm: page not present");
    pa = PTE_ADDR(*pte);
    flags = PTE_FLAGS(*pte);
    if((mem = kalloc()) == 0)
      goto bad;
    memmove(mem, (char*)P2V(pa), PGSIZE);
    if(mappages(d, (void*)i, PGSIZE, V2P(mem), flags) < 0)
      goto bad;
  }
  return d;

bad:
  freevm(d);
  return 0;
}
```

pde_t *pgdir은 parent process의 page table
pde_t *d는 새로운 child process를 위한 page table

새로운 physical page 할당
mem은 새로 할당 받은 page의 virtual address

새로운 physical page에 parent의 physical page를 복사

vm.c

copyuvm()

```
pde_t*
copyuvm(pde_t *pgdir, uint sz)
{
  pde_t *d;
  pte_t *pte;
  uint pa, i, flags;
  char *mem;

  if((d = setupkvm()) == 0)
    return 0;
  for(i = 0; i < sz; i += PGSIZE){
    if((pte = walkpgdir(pgdir, (void *) i, 0)) == 0)
      panic("copyuvm: pte should exist");
    if(!(*pte & PTE_P))
      panic("copyuvm: page not present");
    pa = PTE_ADDR(*pte);
    flags = PTE_FLAGS(*pte);
    if((mem = kalloc()) == 0)
      goto bad;
    memmove(mem, (char*)P2V(pa), PGSIZE);
    if(mappages(d, (void*)i, PGSIZE, V2P(mem), flags) < 0)
      goto bad;
  }
  return d;

bad:
  freevm(d);
  return 0;
}
```

pde_t *pgdir은 parent process의 page table
pde_t *d는 새로운 child process를 위한 page table

virtual address에 대한 page table entry를 child process의 page table에 생성

Project #4. Copy-on-Write

- Implement copy-on-write on xv6

- Implementation details
  1. Modify `copyuvm`() in `vm.c` from copy version to duplicate version

Modify copyuvm()

- 새로운 page 할당없이 parent process의 page를 child process의 mapping table에 매핑
- page table entry(PTE)에 writeable flag (PTE_W) disable
  - 해당 page에 write를 시도하면 page fault 발생 -> Copy-on-write 수행
- physical page의 reference count 증가
- lcr3(V2P(pgdir)) 를 호출해 TLB flush

```
for(i = 0; i < sz; i += PGSIZE){
  if((pte = walkpgdir(pgdir, (void *) i, 0)) == 0)
    panic("copyuvm: pte should exist");
  if(!(*pte & PTE_P))
    panic("copyuvm: page not present");
  pa = PTE_ADDR(*pte);
  flags = PTE_FLAGS(*pte);
  if((mem = kalloc()) == 0)
    goto bad;
  memmove(mem, (char*)P2V(pa), PGSIZE);
  if(mappages(d, (void*)i, PGSIZE, V2P(mem), flags) < 0)
    goto bad;
}
```

Project #4. Copy-on-Write

- Implement copy-on-write on xv6

- Implementation details
  1. Modify `copyuvm`() in `vm.c` from copy version to duplicate version
  2. Managing page reference counter

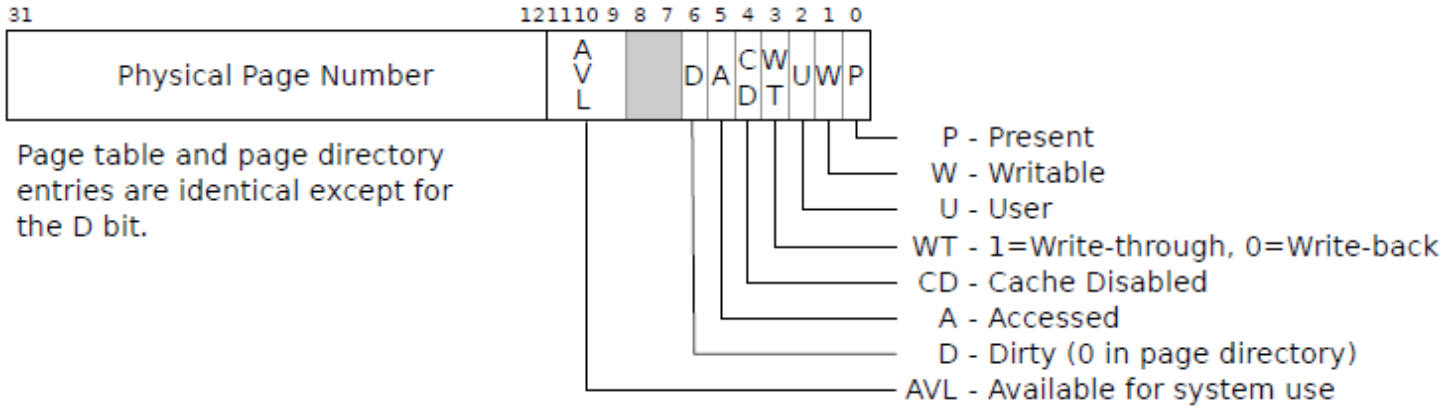
Reference Counter for Physical Pages

- **kalloc.c**
  - 각 physical page의 reference counter를 기록할 배열 생성
    - `uint pgrefcount[PHYSTOP >> PGSHIFT];`
    - `memlayout.h`

```
#define PHYSTOP 0xE000000           // Top physical memory
```

    - `mmu.h`

```
#define PGSHIFT         12          // log2(PGSIZE)
```



Reference Counter for Physical Pages

- **kalloc.c**
  - `void freerange(void *vstart, void *vend)`
    - xv6 commentary book page 33
    - Reference counter 초기화

```
void
freerange(void *vstart, void *vend)
{
  char *p;
  p = (char*)PGROUNDUP((uint)vstart);
  for(; p + PGSIZE <= (char*)vend; p += PGSIZE)
    kfree(p);
}
```

Reference Counter for Physical Pages

- **kalloc.c**
  - void kfree(char *v)
    - reference counter 감소
    - reference counter가 zero가 되면 해당 page를 free page list에 추가

```
void
kfree(char *v)
{
  struct run *r;

  if((uint)v % PGSIZE || v < end || V2P(v) >= PHYSTOP)
    panic("kfree");

  // Fill with junk to catch dangling refs.
  memset(v, 1, PGSIZE);

  if(kmem.use_lock)
    acquire(&kmem.lock);
  numfreepages++;
  r = (struct run*)v;
  r->next = kmem.freelist;
  kmem.freelist = r;
  if(kmem.use_lock)
    release(&kmem.lock);
}
```

Reference Counter for Physical Pages

- **kalloc.c**
  - char* kalloc(void)
    - reference counter 1로 초기화

```
char*
kalloc(void)
{
  struct run *r;

  if(kmem.use_lock)
    acquire(&kmem.lock);
  numfreepages--;
  r = kmem.freelist;
  if(r)
    kmem.freelist = r->next;
  if(kmem.use_lock)
    release(&kmem.lock);
  return (char*)r;
}
```

Reference Counter for Physical Pages

- **`kalloc.c`**
  - Reference counter 관련 API 구현
  - `uint get_refcounter(uint pa)`
    - 인자로 받은 physical address가 속한 physical page의 reference counter를 반환
  - `void dec_refcounter(uint pa)`
    - physical page의 reference counter를 1 감소
  - `void inc_refcounter(uint pa)`
    - physical page의 reference counter를 1 증가

Project #4. Copy-on-Write

- Implement copy-on-write on xv6

- Implementation details
  1. Modify `copyuvm`() in `vm.c` from copy version to duplicate version
  2. Managing page reference counter
  3. Implementing page-fault handler

Page Fault Handler Implememtation

- trap.c

```
void
trap(struct trapframe *tf)
{
  if(tf->trapno == T_SYSCALL){
    if(myproc()->killed)
      exit();
    myproc()->tf = tf;
    syscall();
    if(myproc()->killed)
      exit();
    return;
  }

  switch(tf->trapno){
  case T_PGFLT:
    pagefault();
    break;
  case T_IRQ0 + IRQ_TIMER:
    if(cpuid() == 0){
      acquire(&tickslock);
      ticks++;
      wakeup(&ticks);
      release(&tickslock);
    }
    lapiceoi();
    break;
```

Page Fault Handler Implememtation

- **vm.c**
  - `void pagefault(void)`
    - `rcr2()`를 호출해 page fault가 발생한 virtual address 결정
    - virtual address가 valid한 값인지 확인
    - 해당 virtual address의 page table entry(`pte_t`) 찾기
      - `walkpgdir`()
    - physical address와 reference counter 찾기
      - `get_refcounter()`
    - reference counter가 1보다 큰 경우
      - 새로운 페이지를 할당 받아 기존 페이지를 복사
        - » `kalloc`(), `memmove`()
      - page table entry 내용 변경
      - reference counter 1 감소 (`dec_refcounter()`)
    - reference counter가 1인 경우
      - 현재 page table entry에 writeable flag만 enable
    - `lcr3(V2P(pgdir))`를 호출해 TLB flush

Test programs

- **test1.c**

```
int main(int argc, char **argv)
{
    int before, after;
    int pid;

    printf(1, "TEST1: ");
    before = freemem();
    pid = fork();
    if(pid == 0){
        after = freemem();
        if(before - after == 68)
            printf(1, "OK\n");
        else
            printf(1, "WRONG\n");
        exit();
    }
    else{
        wait();
    }
    exit();
}
```

```
$ test1
TEST1: OK
$ test2
TEST2: OK
$ test3
TEST3: OK
$ test4
TEST4: OK
```

Project #4. Template Code

- **Download xv6-pnu-4.tar.gz from PLATO**
- **Modifications**
  - `freemem()` system call
    - Return the number of free pages in `kmem.freelist`

Submission

- **Compress your xv6 folder as `StudentID-4.tar.gz`**
  - `$tar -czvf StudentID-4.tar.gz ./xv6-pnu-4`
  - Please command `$make clean` before compressing
- **Submit your `tar.gz` file through PLATO**
- **Due date: 6/19 (Fri), 23:59**
- **Late submission penalty**
  - -25% penalty of total mark per day
- **PLEASE DO NOT COPY !!**
  - YOU WILL GET F IF YOU COPIED

Tips

- Reading xv6 commentary will help you a lot
  - https://pdos.csail.mit.edu/6.828/2017/xv6/book-rev10.pdf
    - The line numbers in this book refer to the source booklet below
    - Reading chap. 2 “Page tables” of xv6-commentary will help your project
  - https://pdos.csail.mit.edu/6.828/2017/xv6/xv6-rev10.pdf

- **프로젝트 관련 질문은 PLATO 질의응답 게시판을 활용**