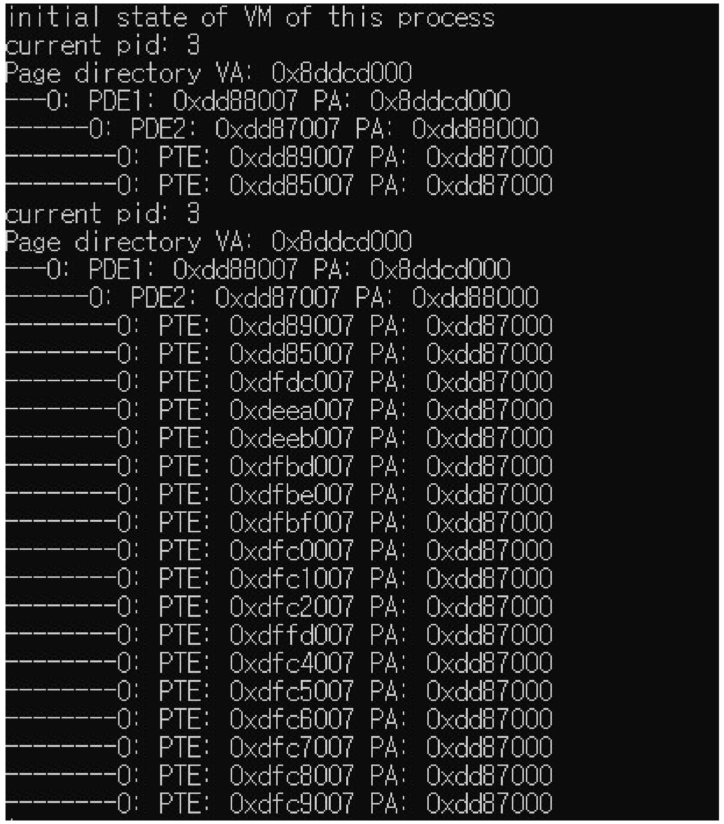
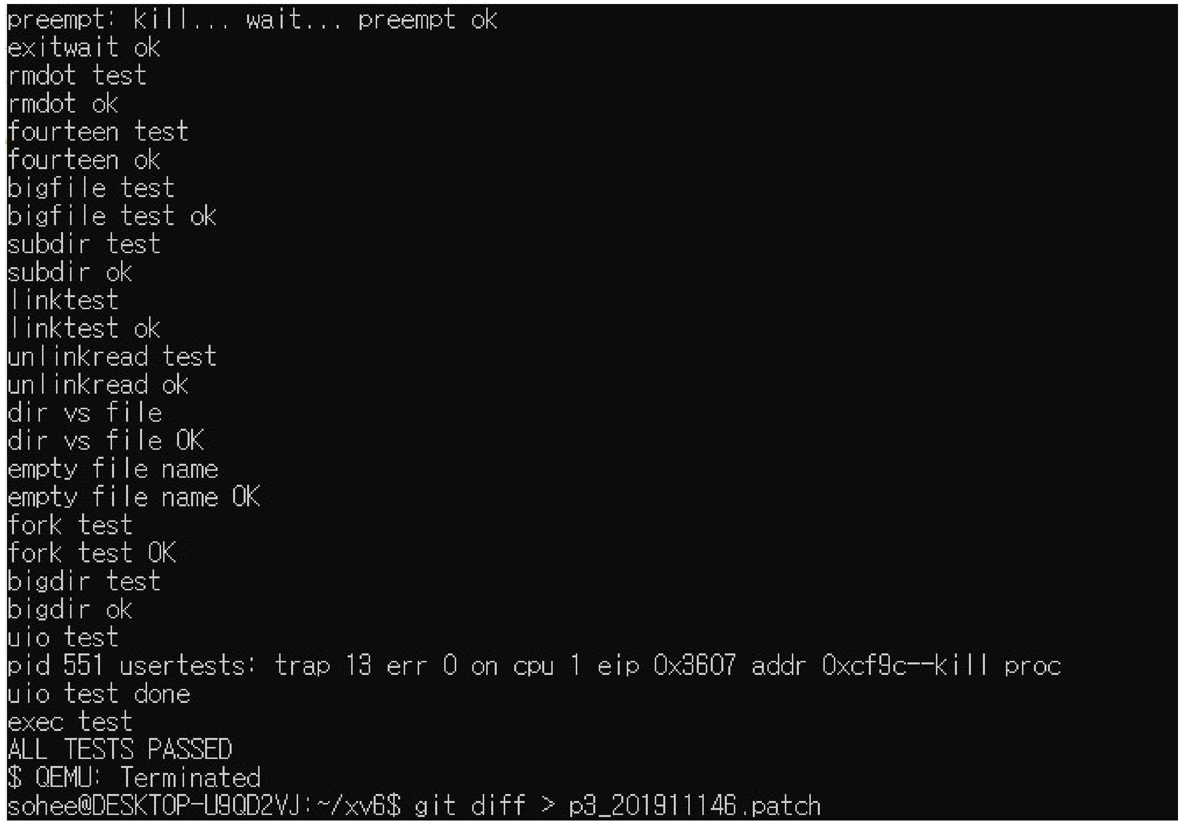
Execution Images

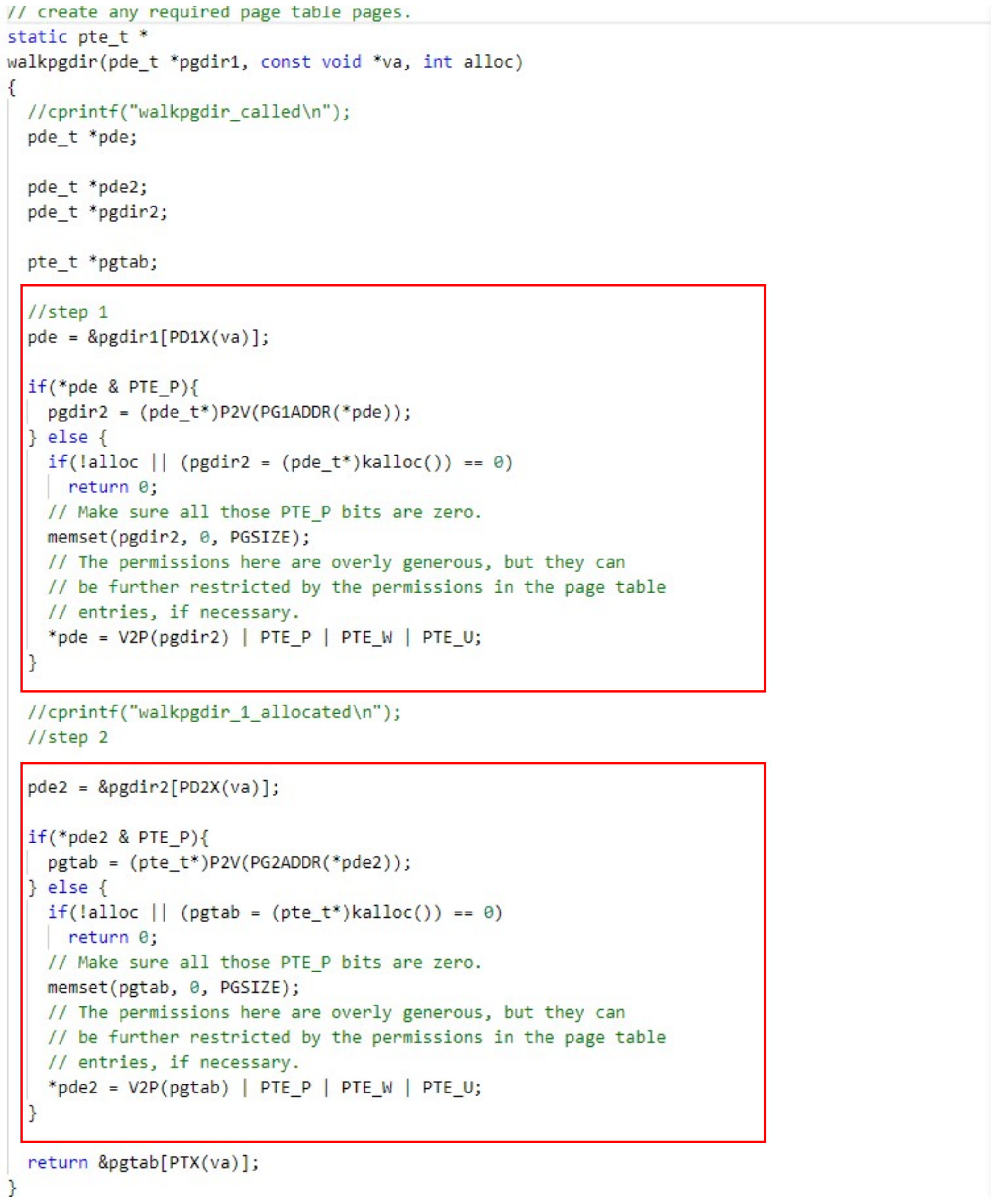


```
initial state of VM of this process
current pid: 3
Page directory VA: 0x8ddcd000
---0: PDE1: 0xdd88007 PA: 0x8ddcd000
------0: PDE2: 0xdd87007 PA: 0xdd88000
--------0: PTE: 0xdd89007 PA: 0xdd87000
--------0: PTE: 0xdd85007 PA: 0xdd87000
current pid: 3
Page directory VA: 0x8ddcd000
---0: PDE1: 0xdd88007 PA: 0x8ddcd000
------0: PDE2: 0xdd87007 PA: 0xdd88000
--------0: PTE: 0xdd89007 PA: 0xdd87000
--------0: PTE: 0xdd85007 PA: 0xdd87000
--------0: PTE: 0xdfdc007 PA: 0xdd87000
--------0: PTE: 0xdeea007 PA: 0xdd87000
--------0: PTE: 0xdeeb007 PA: 0xdd87000
--------0: PTE: 0xdfbd007 PA: 0xdd87000
--------0: PTE: 0xdfbe007 PA: 0xdd87000
--------0: PTE: 0xdfbf007 PA: 0xdd87000
--------0: PTE: 0xdfc0007 PA: 0xdd87000
--------0: PTE: 0xdfc1007 PA: 0xdd87000
--------0: PTE: 0xdfc2007 PA: 0xdd87000
--------0: PTE: 0xdffd007 PA: 0xdd87000
--------0: PTE: 0xdfc4007 PA: 0xdd87000
--------0: PTE: 0xdfc5007 PA: 0xdd87000
--------0: PTE: 0xdfc6007 PA: 0xdd87000
--------0: PTE: 0xdfc7007 PA: 0xdd87000
--------0: PTE: 0xdfc8007 PA: 0xdd87000
--------0: PTE: 0xdfc9007 PA: 0xdd87000
```



```
preempt: kill... wait... preempt ok
exitwait ok
rmdot test
rmdot ok
fourteen test
fourteen ok
bigfile test
bigfile test ok
subdir test
subdir ok
linktest
linktest ok
unlinkread test
unlinkread ok
dir vs file
dir vs file OK
empty file name
empty file name OK
fork test
fork test OK
bigdir test
bigdir ok
uio test
pid 551 usertests: trap 13 err 0 on cpu 1 eip 0x3607 addr 0xcf9c--kill proc
uio test done
exec test
ALL TESTS PASSED
$ QEMU: Terminated
sohee@DESKTOP-U9QD2VJ:~/xv6$ git diff > p3_201911146.patch
```

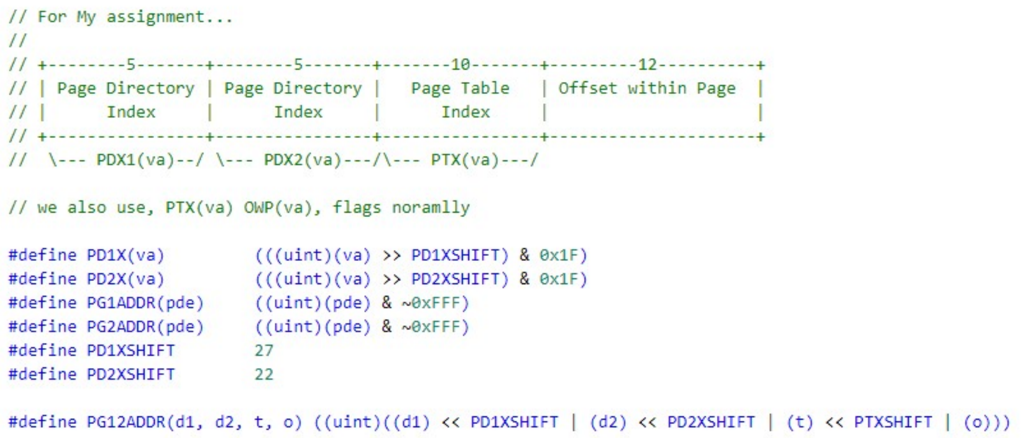
Walkpgdir



```c
// create any required page table pages.
static pte_t *
walkpgdir(pde_t *pgdir1, const void *va, int alloc)
{
  //cprintf("walkpgdir_called\n");
  pde_t *pde;

  pde_t *pde2;
  pde_t *pgdir2;

  pte_t *pgtab;

  //step 1
  pde = &pgdir1[PD1X(va)];

  if(*pde & PTE_P){
    pgdir2 = (pde_t*)P2V(PG1ADDR(*pde));
  } else {
    if(!alloc || (pgdir2 = (pde_t*)kalloc()) == 0)
      return 0;
    // Make sure all those PTE_P bits are zero.
    memset(pgdir2, 0, PGSIZE);
    // The permissions here are overly generous, but they can
    // be further restricted by the permissions in the page table
    // entries, if necessary.
    *pde = V2P(pgdir2) | PTE_P | PTE_W | PTE_U;
  }

  //cprintf("walkpgdir_1_allocated\n");
  //step 2

  pde2 = &pgdir2[PD2X(va)];

  if(*pde2 & PTE_P){
    pgtab = (pte_t*)P2V(PG2ADDR(*pde2));
  } else {
    if(!alloc || (pgtab = (pte_t*)kalloc()) == 0)
      return 0;
    // Make sure all those PTE_P bits are zero.
    memset(pgtab, 0, PGSIZE);
    // The permissions here are overly generous, but they can
    // be further restricted by the permissions in the page table
    // entries, if necessary.
    *pde2 = V2P(pgtab) | PTE_P | PTE_W | PTE_U;
  }

  return &pgtab[PTX(va)];
}
```

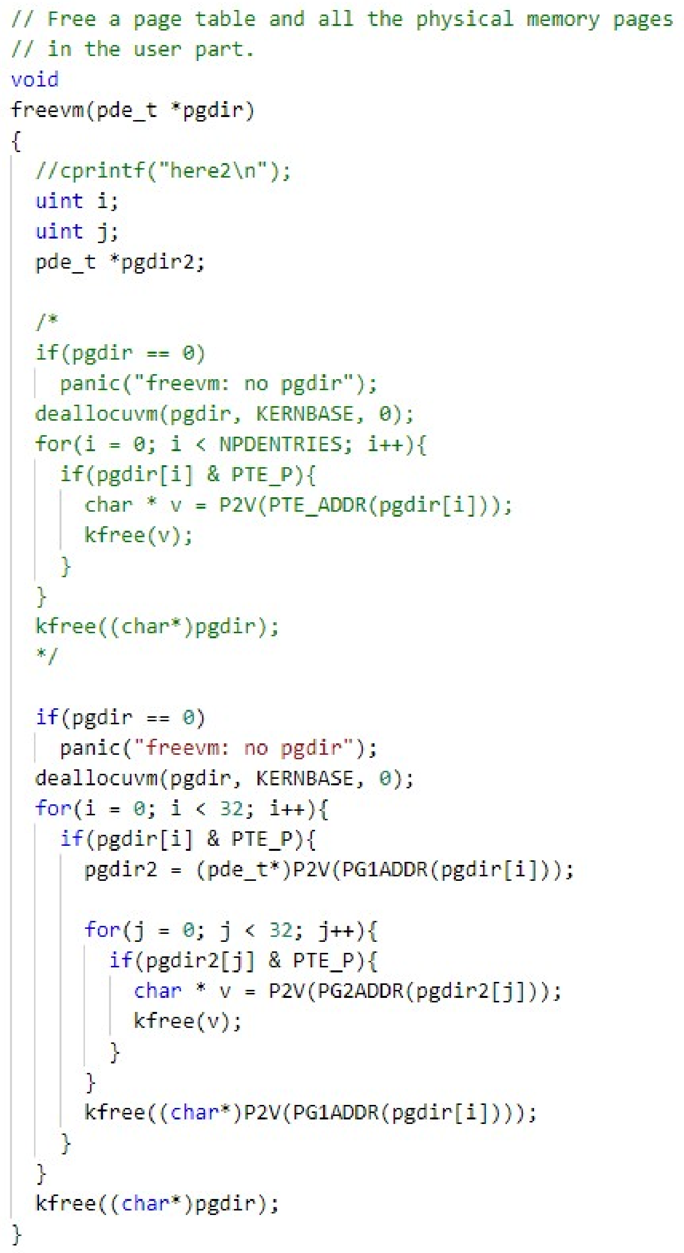
walkpgdir 함수를 3 level paging에 맞도록 수정하였습니다.

매크로는 다음과 같이 변경하여 사용하였습니다.



```c
// For My assignment...
//
// +--------5-------+--------5-------+-------10-------+---------12---------+
// | Page Directory | Page Directory |   Page Table   | Offset within Page |
// |      Index     |      Index     |      Index     |                    |
// +----------------+----------------+----------------+--------------------+
//  \--- PDX1(va)--/ \--- PDX2(va)---/\--- PTX(va)---/

// we also use, PTX(va) OWP(va), flags noramlly

#define PD1X(va)        (((uint)(va) >> PD1XSHIFT) & 0x1F)
#define PD2X(va)        (((uint)(va) >> PD2XSHIFT) & 0x1F)
#define PG1ADDR(pde)    ((uint)(pde) & ~0xFFF)
#define PG2ADDR(pde)    ((uint)(pde) & ~0xFFF)
#define PD1XSHIFT       27
#define PD2XSHIFT       22

#define PG12ADDR(d1, d2, t, o) ((uint)((d1) << PD1XSHIFT | (d2) << PD2XSHIFT | (t) << PTXSHIFT | (o)))
```

PG12ADDR는 Deallocuvm에서 사용될 예정입니다.



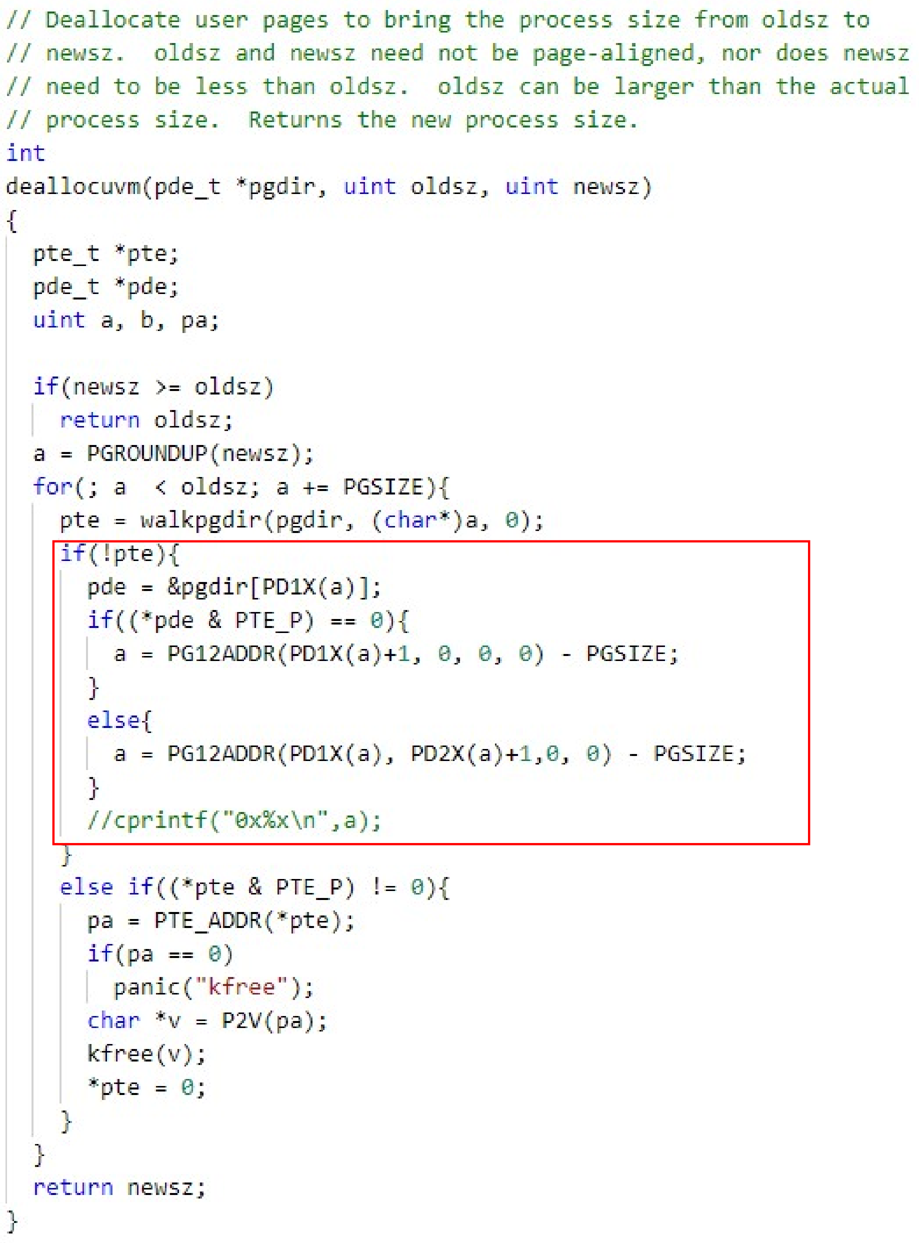
```
// Free a page table and all the physical memory pages
// in the user part.
void
freevm(pde_t *pgdir)
{
  //cprintf("here2\n");
  uint i;
  uint j;
  pde_t *pgdir2;

  /*
  if(pgdir == 0)
    panic("freevm: no pgdir");
  deallocuvm(pgdir, KERNBASE, 0);
  for(i = 0; i < NPDENTRIES; i++){
    if(pgdir[i] & PTE_P){
      char * v = P2V(PTE_ADDR(pgdir[i]));
      kfree(v);
    }
  }
  kfree((char*)pgdir);
  */

  if(pgdir == 0)
    panic("freevm: no pgdir");
  deallocuvm(pgdir, KERNBASE, 0);
  for(i = 0; i < 32; i++){
    if(pgdir[i] & PTE_P){
      pgdir2 = (pde_t*)P2V(PG1ADDR(pgdir[i]));

      for(j = 0; j < 32; j++){
        if(pgdir2[j] & PTE_P){
          char * v = P2V(PG2ADDR(pgdir2[j]));
          kfree(v);
        }
      }
      kfree((char*)P2V(PG1ADDR(pgdir[i])));
    }
  }
  kfree((char*)pgdir);
}
```

freevm

freevm 함수를 3 level paging에 맞도록 수정하였습니다.

두 번의 loop를 사용하였고 원래의 loop는 i<NPDENTRIES 를 사용하지만, 이번에는 2^5 를 사용하기 때문에 i<32 j<32로 바꾸어 주었습니다.

Deallocuvm



```c
// Deallocate user pages to bring the process size from oldsz to
// newsz.  oldsz and newsz need not be page-aligned, nor does newsz
// need to be less than oldsz.  oldsz can be larger than the actual
// process size.  Returns the new process size.
int
deallocuvm(pde_t *pgdir, uint oldsz, uint newsz)
{
  pte_t *pte;
  pde_t *pde;
  uint a, b, pa;

  if(newsz >= oldsz)
    return oldsz;
  a = PGROUNDUP(newsz);
  for(; a  < oldsz; a += PGSIZE){
    pte = walkpgdir(pgdir, (char*)a, 0);
    if(!pte){
      pde = &pgdir[PD1X(a)];
      if((*pde & PTE_P) == 0){
        a = PG12ADDR(PD1X(a)+1, 0, 0, 0) - PGSIZE;
      }
      else{
        a = PG12ADDR(PD1X(a), PD2X(a)+1,0, 0) - PGSIZE;
      }
      //cprintf("0x%x\n",a);
    }
    else if((*pte & PTE_P) != 0){
      pa = PTE_ADDR(*pte);
      if(pa == 0)
        panic("kfree");
      char *v = P2V(pa);
      kfree(v);
      *pte = 0;
    }
  }
  return newsz;
}
```

Deallocuvm 함수를 3 level paging에 맞도록 수정하였습니다.

PTE 가 zero여도 이것이 PD1X가 존재하지 않는 것인지 PD2X가 존재하지 않는 것인지 판별할 수 없습니다.

만약 PD1이 존재하지 않는다면 PD2도 존재하지 않으므로 PD1 의 존재여부를 판단하여

PD1이 존재하지 않는다면 PD1 파트를 +1 해 주고
PD1이 존재한다면 PD2가 존재하지 않는 것으로 판단해서 PD2 파트만 +1 만 해 주면 됩니다.