Manpage for hasPages()

Name

int **hasPages**(int pid);

Description

hasPages is a system call. It will print the address range and pages allocated to the process whose pid matching the input pid. The output addresses are virtual addresses.

Input Value

pid , the pid of a running process

Return Value

1, if succeed

Error

-1, if the input is not valid, pid should be a int bigger than 0

-2, the no process running has the pid input.

Man page for myV2p()

Name

int **myV2p**(int Vp,int operation);

Description

myV2p is a system call. It takes a virtual address on the memory space in the caller process and a operation, either read or write. This system call first try to find the PTE of that address, if it valid then loop up the page table and find the physical address.

Input Value

Vp is a virtual address

Operation:0 if read that address, 1 if write that address

Return Value

The physical address for the input virtual address, if succeed

Error

-1, if the input is not valid, operation should only be 0 or 1

-2, if the address is an invalid virtual address, i.e. can not find the corresponding PTE

-4, if the address is not writable, this returned when used in write operation

Design of the system calls

myV2p() is implemented in sysproc.c

pde = &pgdir[PDX(Vp)];
//*pde points to a physical_location where we can find the pte,
pgtab = (pte_t*)P2V(PTE_ADDR(*pde));

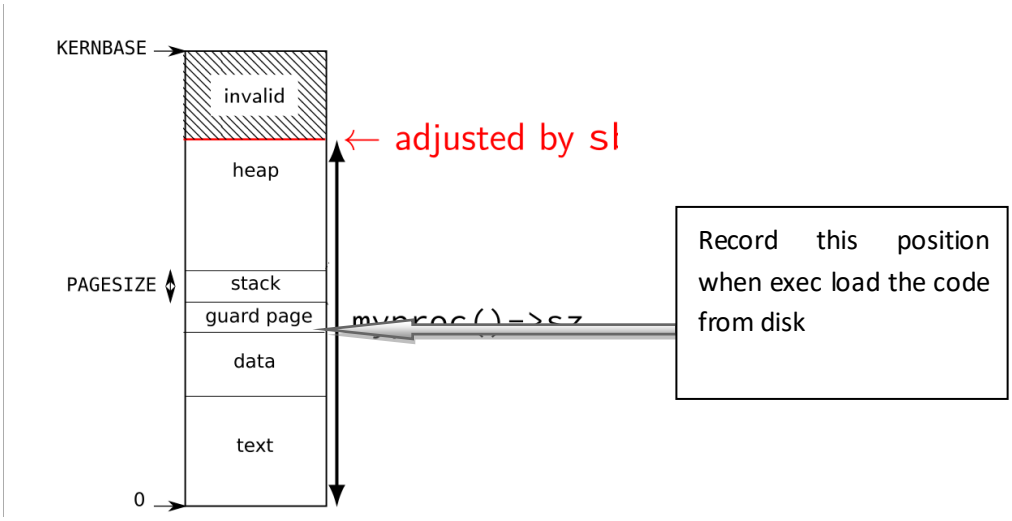
Page table is located in pgtab,
pte = &pgtab[PTX(Vp)];// this pte points to the physical page table entry.

Use *pte get the physical table and add the offset, it gives us the physical address.

physical_address=(int)(PTE_ADDR(*pte))+(Vp&0x3FF);

Design of the system calls

hasPages() is implemented in proc.c



As in textbook, store the pages used when exec()load the code from disk, this is the space used for data and test. The one page guard page, and one page for stack. Stack page is always one page. Then we can use sz in proc get the top of the heap, with the end of stack, we can find out pages for heap.