Write the source code commentary of the following routines in xv6.

• walkpgdir

This function returns the address of the PTE. First of all this function gets the base address and checks if it is valid or not. If it is valid, it puts the address of the subtable in pgtab. If the index address wasn't valid then it creates a new subtable and sets all values of pgtab to zero. Then it points the index address to pgtab and marks it valid, writeable and can be modified by the user. Then it finally returns the address of the required row in pgtab.

• mappages

This function maps each of the virtual addresses to physical addresses. Basically, it creates a Page Table Entry for each virtual address. This PTE at a virtual address would refer to a physical address.

First of all this function creates variables to go through all the virtual addresses. *last is to check whether it has reached the end of the page table or not.

It goes through all the virtual addresses, so in each iteration, it calls walkpgdir which returns address of the PTE for this virtual address.

After walkpgdir returns, it initializes the PTE. This PTE now gets a page number in physical space, the required permissions and PTE_P to make the PTE valid.

It also gives a kernel panic if that pte is already used before initializing.

• setupkvm

kvmalloc calls setupkvm in order to create a page directory using kmap.

setupkvm first uses kalloc to allocate a page of pointer to save page directories. These pointers point to a page of the page table.

setupkvm calls mappages for every virtual to physical address mapping in kmap (constructing the page directory entries)

As mentioned above, mappages covers virtual address space in 4KB pages and locates PTE via walkpgdir function.

• allocuvm

Called by exec to allocate memory for every ELF segment. Takes arguments old size and new size and increases size of process to new size, returning new size if successful and 0 otherwise.

• loaduvm
Called by exec to load each ELF segment into memory by reading the memory executable from disk using the readi function.

Takes arguments sz, ip and addr, and loads program section of size sz from ip file to address add and returns 0 if successful and -1 otherwise.

• uva2ka

This function maps user virtual address to kernel address. It first calls walkpgdir(), to get the page table entry(PTE) of uva, according to given page directory. It then checks whether the memory that this PTE refers to is valid and also modifiable by a user program, otherwise, it returns. It then returns the virtual address of this PTE (as this is in kernel space, it doesn't matter according to which user program).

• freevm

This function is used to free up the page table and all its physical memory pages of a user program.

It takes the page directory of the user program as input. It panics the kernal if no page directory is given.

It first calls deallocuvm, with KERNBASE as old_size,(since all the memory in a user programs VA, from 0 to KERNBASE is specific to the program, and thus can be cleaned up.), and 0 as new size. This calls frees the physical memory pages.

It then iterates over entries in first level of the page directory, cleaning all the pages that formed the 2nd level page tables.

It finally frees the 1st level page table.