OS-344

Assignment-3

Doubt Clearing session

Presented by

Amit Puri

Teaching Assistant

Focus On

1. Memory Management
   a. Lazy Memory Allocation
   b. Disk swapping of user process Pages

Lazy Memory Allocation

• Delaying the memory allocation of an object until the time of actual need of that object.

sbrk() in sysproc.c

```
int
sys_sbrk(void)
{
 int addr;
  int n;


  if(argint(0, &n) < 0)
    return -1;
  addr = myproc()->sz;
  myproc()->sz += n;

    //if(growproc(n) < 0)
      //return -1;
  return addr;
}
```

Lazy Memory Allocation

• Delaying the memory allocation of an object until the time of actual need of that object.

sbrk() in sysproc.c

```
int
sys_sbrk(void)
{
 int addr;
  int n;


  if(argint(0, &n) < 0)
    return -1;
  addr = myproc()->sz;
  myproc()->sz += n;

    //if(growproc(n) < 0)
      //return -1;
  return addr;
}
```

• How to know, when process needs any new resource in the memory?

On trap due to Pagefault

tf->trapno=T_PGFLT

Lazy Memory Allocation

• Delaying the memory allocation of an object until the time of actual need of that object.

sbrk() in sysproc.c

```
int
sys_sbrk(void)
{
 int addr;
  int n;


  if(argint(0, &n) < 0)
    return -1;
  addr = myproc()->sz;
  myproc()->sz += n;

    //if(growproc(n) < 0)
      //return -1;
  return addr;
}
```

Reusing Code

allocuvm()

Used when a process needs to grow its memory size.

And reuse some other functions called by this.

Disk Swapping

- Swapping out the pages to secondary storage to make it look like that we have a bigger memory.
- Backup a page with unique name as <pid>_<VA[20:]>
- Kernel Process
  - void create_kernel_process(const char *name, void (*entrypoint)());
- Two such processes are required one for swap in and other for swap out.
- These process should be part of process table and should have init() as parent
- It should never return to userspace and end only in kernel space at completion.

*** You might need to implement these functions in Proc.c**

Disk Swapping

- Swap out (when there is not enough memory for a process)
  - Suspend (Sleeping state)
  - Send a request to kernel swap out process
  - Select a page using LRU and save its contents in the disk.
  - Remove the corresponding page using bit flag.
  - Mark it a free page.

* Requests should be handled via a queue and wake-up all or one of the process on having a free physical page(whichever seems easy to you) and everytime suspend the process after use.

Disk Swap in

- Swap in (when there is a page fault due to swapped out page)
  - Suspend the process and send a request to kernel swap in process.
  - Allocate a single page in memory.
  - Read page from disk and fill it in memory.
  - Map to the virtual address.

* Requests should be handled via a queue, suspend the kernel process after execution and wake-up corresponding process after swapping-in

Testing

- Multiple child processes each requiring multiple pages in memory, so that it may not be served using available number of free pages.
- If this user code run successfully that means disk swapping is working fine.
- Using PHYSTOP in memlayout.h, end of physical memory can be decreased while testing.

Questions?