Assignment 1:

proc\_list -> 1 2 X 4 5, handle?

struct process\_struct;

pid

pointer to heap

an array of active processes: process\_struct[MAX\_SIZE]

struct heap;

size

type

heap array

static int active\_cnt=0;

init\_module():

create entry

pointers register(file\_operations)

malloc space to \*process\_struct[MAX\_LIMIT]; // initialize NULL

locks init

get\_idx\_from\_pid(pid):

return idx or -1

open():

find pid?

check if\_already open: O(N) time and < MAX\_LIMIT

malloc new process\_struct

active = 1

heap init

close():

free heap

free process\_struct[idx]

write():

find pid?

"user 2" tries here, acquire, spinlock, semaphore

lock

here control, switch, "user 1" here.

unlock!

read():

same as write()

Assignment 2:

All the other structs are the same

struct heap;

size

type

heap array

last\_inserted

init\_module():

<Add>

file\_ops.unlocked\_ioctl = ioctl;

copy open()

copy close()

file\_ioctl():

switch case

WRITE:

!lock

copy\_from\_user

handler()

!unlock

return err

READ:

!lock

obj --> pointer

err = handler()

if(!err and !copy\_to\_user())

!unlock

return err

handle\_pb2\_set\_type(struct):

check state

handle\_pb2\_insert(int):

check state

handle\_pb2\_get\_info(struct):

check state

handle\_pb2\_extract(struct):

check state

To Confirm:

* Locks? whole ioctl function or indiv. cases [Threads/Processes]
* open and close!!!! Handle concurrency