Lab 2: time-shared multiprogramming

First task: allow the user to compile/run programs that use malloc() + standard I/O library

- ioctl() (kinda)
- fstat() (kinda)
- sbrk()
- ...

Second task: implement system calls for a simple shell

- fork()
- execve()
- wait()
- exit()

Third task: implement process IDs

- getpid()
- getppid()

Fourth task: make time slicing work

Lab 2—more on fork(), execve(), and wait()

- How I split my memory up
  - Maintain a global partition map to keep track of what parts are allocated/free
  - Write two helper functions:
    - int alloc_mem_part(int *base, int *limit);
    - void free_mem_part(int base);
  - Allocate memory inside init_user_process() and fork()
  - Free memory inside exit() system call
- How to tackle execve()
  - Start by factoring out part of init_user_proc()
  - perform_exec(struct PCB_struct *pcb, char *fname, char **argv)
    - This is where you do:
      - load_user_program()
      - MoveArgsToStack()
      - InitCRuntime()
    - You can reuse this when implementing execve()
  - Ignore envp[] — we don't have environment variables
- How to tackle fork()
  - Create a new PCB and allocate some memory for it
  - The new PCB should have the same register values as its parent, but a different base and limit
  - After fork() finishes, there should be one additional PCB on the ready queue
    - Queue both parent + child

The wait() system call

- Each PCB must keep track of who its parent is
  - Add a pointer field that points to the parent PCB
- The very first process must be a child of the Init PCB
  - Init never gets run; it only exists so that the first PCB has a parent
- Each PCB must also keep track of all the processes that are waiting on it
  - You could use a doubly-linked list for this
  - When a child calls exit(), it will be added to its parent's waiting list
  - wait() will clean up one child or block if no children are ready to be cleaned up yet
- When a process dies, its children become children of Init
- Modify the PCB struct to keep track of all its active children
  - When fork() is called, the new child must be added to its parent's active children list
  - When exit() is called, all active children must be moved to be children of Init

The exit() system call

- What do I do when I exit?
  - Record my exit status in the PCB
  - Free the memory partition I used
  - If I have any children, make them all children of Init
  - Delete myself from my parent's list of children
  - Add myself to my parent's list of waiters
  - Unblock my parent's wait semaphore (if my parent isn't Init)
  - kt_exit() and we're done!

The sbrk(), fstat(), and ioctl() system calls

- sbrk() increments the location of the program break
  - The division between the heap and unallocated memory
  - Return the location of the program break before incrementing (yeah it's weird...)
- We only implement one case of ioctl() for this lab
  - Use ioctl_console_fill()
- For fstat(), use stat_buf_fill()
  - We only care about file descriptors 0, 1, and 2
    - fd 0 → size = 1
    - fd 1, 2 → size = 256
- The man pages are your best friend!

Process IDs

- Process IDs will be of type unsigned short
- How do we ensure distinct process IDs?
    - You can use a red-black tree!
        - /cs/faculty/rich/cs170/include/jrb.h

Let's talk about ksh…

- If ksh works, great!
- If it doesn't, don't stress
    - Either way, please write your own test codes to test fork(), execve(), wait(), etc.
- We will NOT grade your lab 2 implementation against ksh, so don't let it distract you

Time slicing

- This lets us switch between processes!
- Just call start_timer(ticks) once
    - I did it at the end of InitUserProcess()
    - You can use ticks = 10 (or whatever, really)
- Simulator throws a TimerInt every time interval
    - When handling this, just put the current running process at the end of your ready queue

PCB struct additions

```
struct PCB_struct {
    int mem_base;
    int mem_limit;
    int data_end;
    int sbrk;
    unsigned short pid;
    struct PCB_struct *parent;
    kt_sem *waiter_sem;
    Dllist waiters;
    Rb_node children;
    int exit_status;
    int registers[NumTotalRegs];
};
```

Some other system calls

- close()
  - Return (-1 * EBADF)
- getdtablesize()
  - Return the value 64
- getpagesize()
  - Return the value of PageSize in simulator.h (512)