Project 2 Roadmap++

Awwww! ugly slides.
**Review**

Process A

```plaintext
main() {
    for(1000) {
        Print "A"
    }
    Kill(B, SIGUSR1)
}
```

Process B

```plaintext
function handler() {
    Print "HANDLING"
}

main() {
    Signal(&handler, SIGUSR1)
    for(;;) {
        Print "B"
    }
}
```

**Output**

```
AABBAABBAABB.........HANDLING BBBBBBBBBBBBBBBBBBBB.....
```

1000 A’S
System Calls

- **Sys_Signal**: register a signal handler
- **Sys_Kill**: send a signal
- **Sys_RegDeliver**: initialize signal handling for a process
- **Sys_WaitNoPID**: wait for any child process to die
- **Sys_ReturnSignal**: indicate completion of signal handler
System Calls

- **Sys_Signal:** register a signal handler
- **Sys_Kill:** send a signal
- **Sys_RegDeliver:** initialize signal handling for a process
- **Sys_WaitNoPID:** wait for any child process to die
- **Sys_ReturnSignal:** indicate completion of signal handler

Referenced in user code
Process A

```c
main() {
    for(1000) {
        Print "A"
    }
    Kill(B, SIGUSR1)
}
```

Process B

```c
function handler() {
    Print "HANDLING"
}

main() {
    Signal(&handler, SIGUSR1)
    for(;;) {
        Print "B"
    }
}
```

Output

```
AABBAABBAABB........HANDLING BBBBBBBBBBBBBBBBBB.....
```

1000 A’S
System Calls

- **Sys_Signal:** register a signal handler
- **Sys_Kill:** send a signal
- **Sys_RegDeliver:** initialize signal handling for a process
- **Sys_WaitNoPID:** wait for any child process to die
- **Sys_ReturnSignal:** indicate completion of signal handler
System Calls

- **Sys_Signal:** register a signal handler
- **Sys_Kill:** send a signal
- **Sys_RegDeliver:** initialize signal handling for a process
- **Sys_WaitNoPID:** wait for any child process to die
- **Sys_ReturnSignal:**
  - Executed by stub code once a signal has been handled
Helper Functions

- Send_Signal
- Set_Handler
- Check_Pending_Signal
- Setup_Frame
- Complete_Handler
Process A
main() {
    for(1000)
        Print “A”
    Kill(B, SIGUSR1)
}

Process B
function handler() {
    Print “HANDLING”
}

main() {
    Signal(&handler, SIGUSR1)
    for(;;)
        Print “B”
}
Overview

A

B
Overview

A

RegDeliver
Signal(SIGCHIL)

B

RegDeliver
Signal(SIGCHIL, ....)
Helper Functions

- Send_Signal
- Set_Handler
- Check_Pending_Signal
- Setup_Frame
- Complete_Handler

Look at Scheduler
Scheduler: w/o signals

scheduler

A

B

src/geekos/lowlevel.asm
Scheduler: w/ signals
Scheduler: w/ signals

A

scheduler

Check Pending Signal

B

B's user level sig. hand.
Check Pending Signal

- Boolean output
- Determines whether to proceed with signal handling
Scheduler: w/ signals

A

scheduler

B

Check

Pending Signal

ture

B's user level sig. hand.

false
Scheduler: w/ signals

A

scheduler

Check Pending Signal

Setup Frame

B’s user level sig. hand.

B
Setup Frame

- Sets up state to enable user-level handling code execution
Scheduler: w/ signals

A

scheduler

Pending Signal

Check

Setup Frame

B’s user level sig. hand.

B

B
Setup Frame

- Sets up state to enable user-level handling code execution
- How are functions called?
Function Calls

- Parameter of return address is stored on the stack so when finished
  - Pop off stack
  - Continue execution

- Setup Frame
  - Enables user stack to keep:
    - Interrupt State Vector
    - Return address
Storing Return Address

- Want complete_handler to execute once user level handling done.
- Hack (well kind of)
  - Place address of return_signal as return address on stack
  - Now return_signal stored as function
Scheduler: w/ signals

A

scheduler

Check
Pending
Signal

Setup
Frame

B's
user
level
sig.
hand.

Completer
h Handler

B