Additional slides

Nonlocal Jumps: `setjmp/longjmp`

- **Powerful (but dangerous) user-level mechanism for transferring control to an arbitrary location**
  - Controlled to way to break the procedure call / return discipline
  - Useful for error recovery and signal handling

- `int setjmp(jmp_buf j)`
  - Must be called before longjmp
  - Identifies a return site for a subsequent longjmp
  - Called **once**, returns **one or more** times

- **Implementation:**
  - Remember where you are by storing the current *register context*, *stack pointer*, and *PC value* in `jmp_buf`
  - Return 0

`setjmp/longjmp`(cont)

- **`void longjmp(jmp_buf j, int i)`**
  - Meaning:
    - return from the **`setjmp`** remembered by jump buffer **`j`** again …
    - … this time returning **`i`** instead of 0
  - Called after **`setjmp`**
  - Called **once**, but **never** returns

- **`longjmp` Implementation:**
  - Restore register context (stack pointer, base pointer, PC value) from jump buffer **`j`**
  - Set %**`eax`** (the return value) to **`i`**
  - Jump to the location indicated by the PC stored in jump buf **`j`**

`setjmp/longjmp` Example

- **Goal: return directly to original caller from a deeply-nested function**

```c
/* Deeply nested function foo */
void foo(void)
{
    if (error1)
        longjmp(buf, 1);
    bar();
}

void bar(void)
{
    if (error2)
         longjmp(buf, 2);
}
```

setjmp/longjmp Example (cont)

```c
jmp_buf buf;

int error1 = 0;
int error2 = 1;

void foo(void), bar(void);

int main()
{
    switch(setjmp(buf)) {
    case 0:
        foo();
        break;
    case 1:
        printf("Detected an error1 condition in foo\n");
        break;
    case 2:
        printf("Detected an error2 condition in foo\n");
        break;
    default:
        printf("Unknown error condition in foo\n");
    }
    exit(0);
}
```

Limitations of Nonlocal Jumps

- **Works within stack discipline**
  - Can only long jump to environment of function that has been called but not yet completed

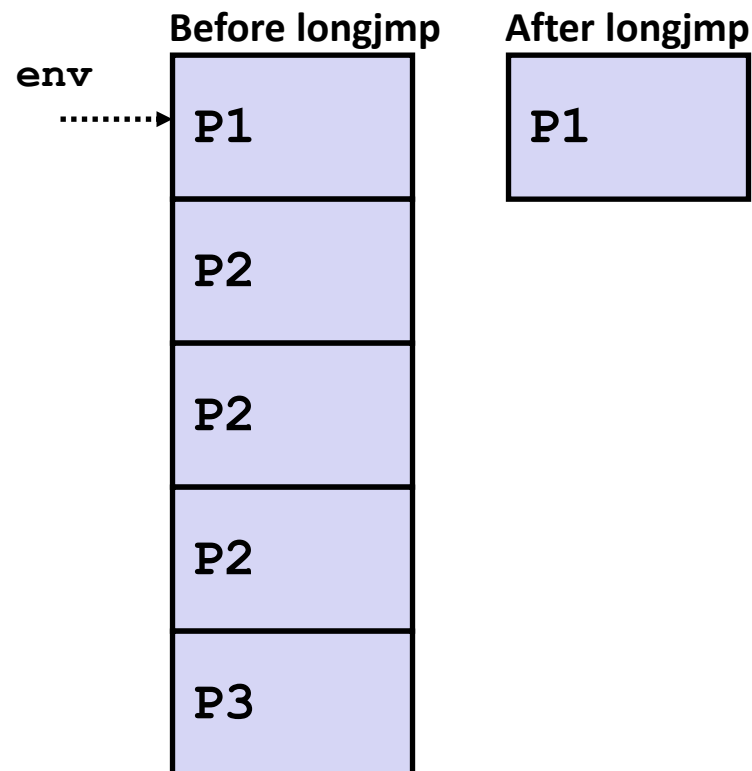**Before longjmp**   **After longjmp**

env

```
jmp_buf env;

P1()
{
  if (setjmp(env)) {
    /* Long Jump to here */
  } else {
    P2();
  }
}

P2()
{  . . . P2(); . . . P3(); }

P3()
{
  longjmp(env, 1);
}
```

Before longjmp
P1
P2
P2
P2
P3

After longjmp
P1

Limitations of Long Jumps (cont.)

- **Works within stack discipline**
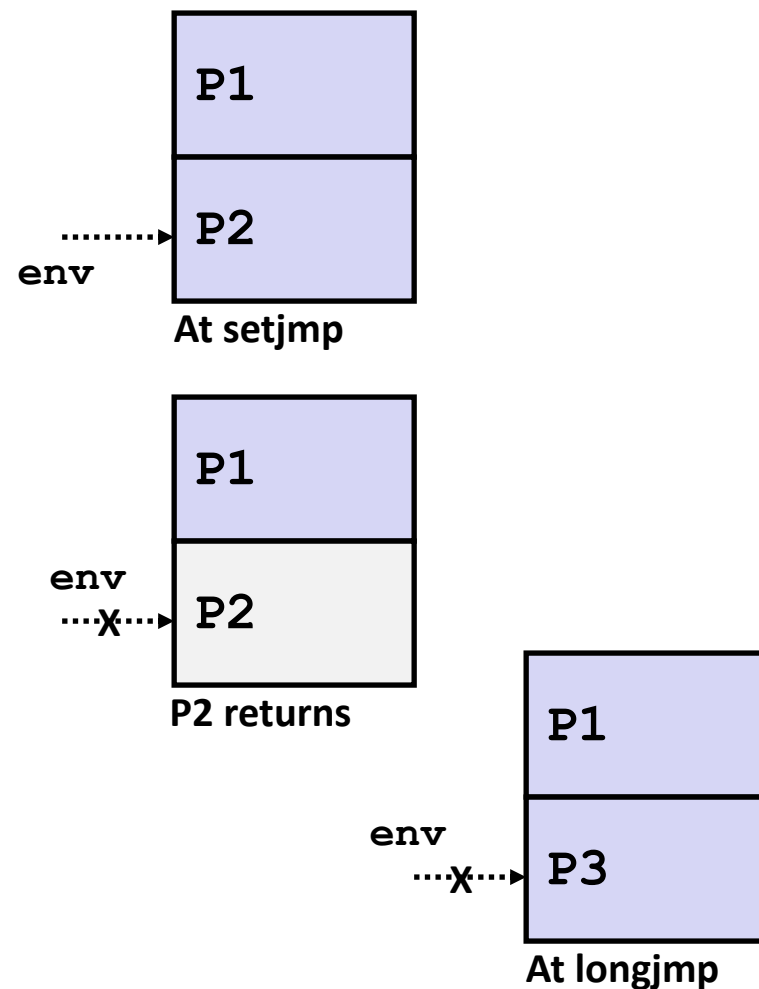 - Can only long jump to environment of function that has been called but not yet completed

```
jmp_buf env;

P1()
{
  P2(); P3();
}


P2()
{
  if (setjmp(env)) {
    /* Long Jump to here */
  }
}


P3()
{
  longjmp(env, 1);
}
```


env ....► P2

At setjmp

env ....x....► P2

P2 returns

env ....x....► P3

At longjmp

Putting It All Together: A Program That Restarts Itself When `ctrl-c'd`

```c
#include "csapp.h"

sigjmp_buf buf;

void handler(int sig)
{
    siglongjmp(buf, 1);
}

int main()
{
    if (!sigsetjmp(buf, 1)) {
        Signal(SIGINT, handler);
         Sio_puts("starting\n");
    }
    else
        Sio_puts("restarting\n");

    while(1) {
        Sleep(1);
        Sio_puts("processing...\n");
    }
    exit(0); /* Control never reaches here */
}
```
restart.c

```
greatwhite> ./restart
starting
processing...
processing...
processing...
restarting
processing...          ⟵ Ctrl-c
processing...
restarting
processing.            ⟵ Ctrl-c
processing...
processing...
```