switch Statement
Switch versus if/else

- Subset of functionality of if/else
- Works in cases with one value to test
- Switches can be more compact
  - No need to retype variable being tested
  - Range syntax (for GCC-compatible compilers)
  - Easier to read when used properly
- Good when combined with enums
switch Statement

With enums

Example

typedef enum {
    PARAM_EVENT_NONE,
    PARAM_EVENT_REQUEST_LIST_RECEIVED,
    PARAM_EVENT_REQUEST_READ_RECEIVED,
    PARAM_EVENT_SET_RECEIVED
} ParamEvent;

Example

ParamEvent x;
switch (x) {
    case PARAM_EVENT_NONE:
        puts("PARAM_EVENT_NONE found.");
        break;
    case PARAM_EVENT_REQUEST_LIST_RECEIVED:
        puts("PARAM_EVENT_REQUEST_LIST_RECEIVED found.");
        break;
};
switch Statement

With enums

Example

```c
ParamEvent x;
switch (x) {
    case PARAM_EVENT_NONE:
        puts("PARAM_EVENT_NONE found.");
        break;
    case PARAM_EVENT_REQUEST_LIST_RECEIVED:
        puts("PARAM_EVENT_REQUEST_LIST_RECEIVED found.");
        break;
}
```

Errors

test.c:141:1: warning: enumeration value
'PARAM_EVENT_REQUEST_READ_RECEIVED' not handled in switch
test.c:141:1: warning: enumeration value
'PARAM_EVENT_SET_RECEIVED' not handled in switch
switch Statement

With enums

Example

```c
ParamEvent x;
switch (x) {
    case PARAM_EVENT_NONE:
        puts("PARAM_EVENT_NONE found.");
        break;
    case PARAM_EVENT_REQUEST_LIST_RECEIVED:
        puts("PARAM_EVENT_REQUEST_LIST_RECEIVED found.");
        break;
    default:
        break;  // FATAL_ERROR()
}
```
switch Statement

Local variables

Example

```c
int x; int i;
switch (x) {
    case 3:
        int i;
        for (i = 0; i < 3; ++i) {
            puts("x");
        }
        break;
    ...
}
```

Errors

**error:** a label can only be part of a statement and a declaration is not a statement
switch Statement

Local variables

Example

```c
int x;
switch (x) {
    case 3:
        
        int i;
        for (i = 0; i < 3; ++i) {
            puts("x");
        }

        break;

    ...
}
```
State machines
Serial
115200 bits/second
State

down

1 2 3 4

up

1 2 3 4
State machines

- Known as Finite State Machines (FSM)
- Mathematical model of computation where system has a single state
- Triggering conditions can change that state
- FSMs are defined completely by both their states and the transitions between them
State machines

State

- The system only exists in one state at a time
- State persists through time
- Certain conditions can change the state to another state
  - These are specific to the current state
State machines

Transitions

• Events trigger transitions between states
• A combination of events can be used
• Transitions are all mutually exclusive
• At any given time there must be a valid transition for a state
  – If no transition is explicitly stated, an implied loopback transition exists
State machines

Benefits

• Provides a formal way to reason about a system
  – Allows for testing before writing any code
• Can be easily visualized
• Are language independent
• States are only dependent on current state and current inputs
State machines
When to use

- Can be used whenever there are a finite set of states for the system
  - Car transmission
  - Stoplight
  - Vending machine
  - Toaster oven
  - Video games
State machines

Use in the SeaSlug

• Transmission protocol
  – Mission management
  – Parameter management
• Operating state
  – Handling errors/system faults
• Calibration
  – Rudder
  – Radio controller
3-digit

Locked

Unlock

Unlocked

one

right

4

2

right

3

entered

4

entered

Option:

1 2 3
4 5 6
7 8 9
Lock

3 1 4
```c
typedef enum { STATE_1, STATE_2 } SystemState;
static SystemState state;
{
    switch (state) {
    case STATE_1:
        // State 1
        if (condition1) {
            Action1();
            state = STATE_2;
        }
        break;
    case STATE_2:
        // State 2
        if (condition2) {
            Action2();
            state = STATE_1;
        }
    }
}
```
State machines

Integrating

Example

typedef enum { STATE_1, STATE_2 } SystemState;
static SystemState state;
int main (void) {
    // Initialize system

    // Event loop
    while (1) {
        // State machine
        switch (state) {
            ...
        }
    }
}
State machines
Bounce lab example

• Live coding example!
State machines
Bounce lab example

- State machines rely on conditions to trigger state machines
  - Fits in nicely with event-driven programming
- Use an enum datatype for your states and a switch-statement to check them
  - Provides some compile-time checks
State machines
Bounce lab example

• Live coding example!