
Version 1.01
UCSD CSE 30 Section B
Computer Organization and Systems Programming
PA 3 Prep
Keith Muller
DEC PDP 11/45 - 1973

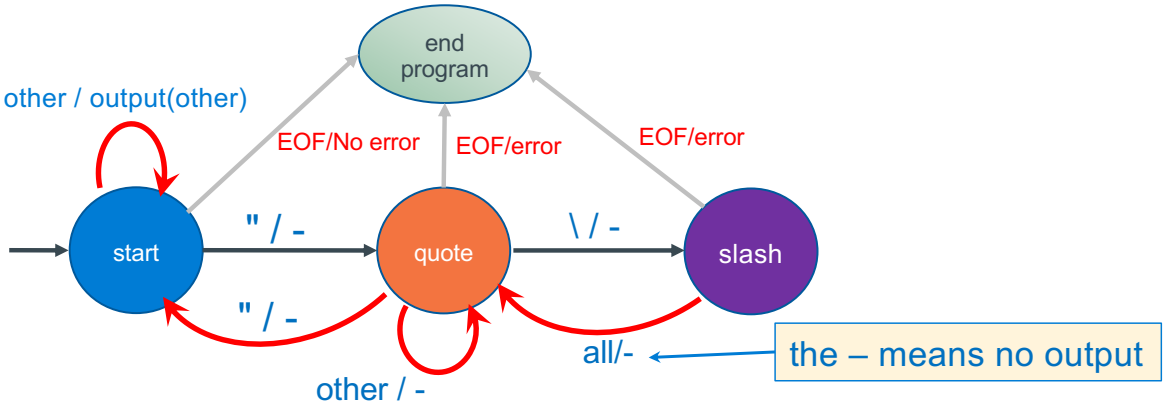
PA3: Programming a Deterministic Finite Automaton

Rules for this DFA example in these slides
Copy input to output while removing everything in "strings" from output

input: ab"foo"cd
output: abcd

Special Case: If Inside a string, a \ is an escape sequence, ignore the next char
Allows you to put an " in a string

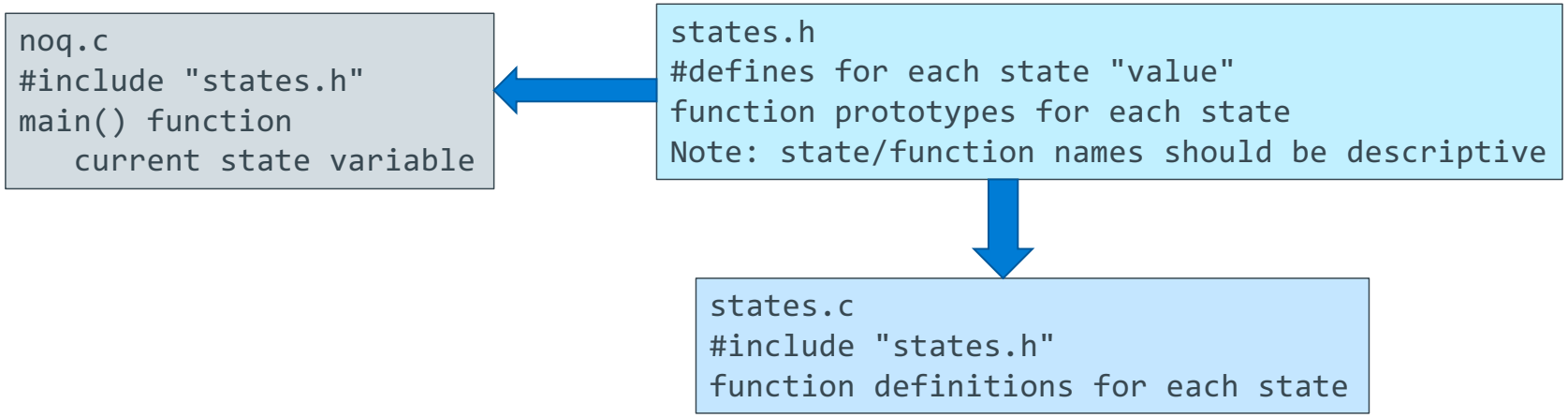
input: ab"foo\"bar"cd
output: abcd



x

Programming a Deterministic Finite Automaton – The Files

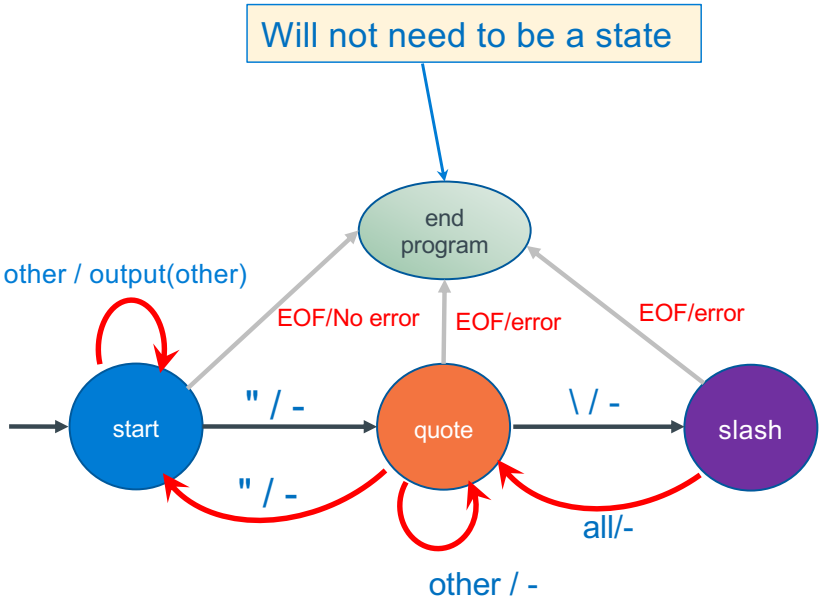
- Break the program into three files
- `noq.c` is where main loop is, imports declarations in `states.h`
- `states.h` is the public interface to the state handlers in `states.c`
- `states.c` definition of the state handler functions, imports declarations in `states.h`
- Observe there is no .h file for noq.c, as it does not have any interface exports



x

Programming a Deterministic Finite Automaton - states.h

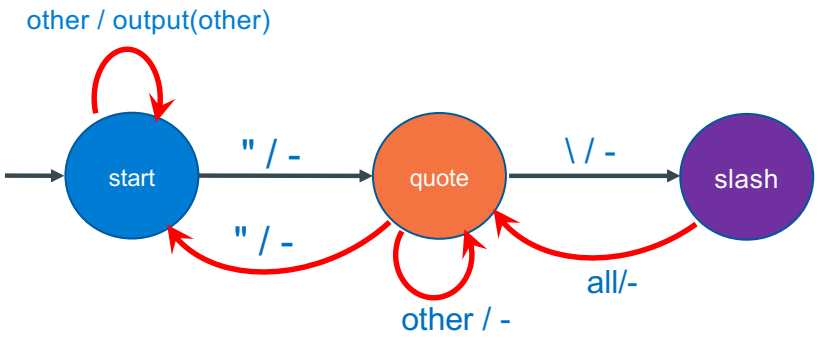
```
// public interface file states.h
#ifndef STATES_H
#define STATES_H

// Assign a value for each state
#define START   0
#define QUOTE   1
#define SLASH   2

// Function prototypes
// for each state handler
int STARTstate(int);
int QUOTEstate(int);
int SLASHstate();

#endif
```



- Each function implements the **arcs** out of that state
  1. **returns the next state** based on the input
  2. **performs** any **actions associated** with **arc taken**

x

Programming a Deterministic Finite Automaton – states.c



states.c

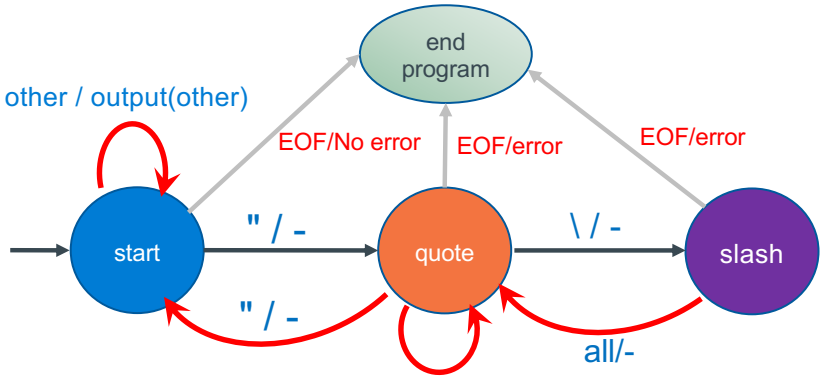
```
#include <stdio.h>
#include "states.h"
int STARTstate(int c)
{
    if (c == '"')
        return QUOTE;       // saw a double quote
    putchar(c);             // echo input
    return START;           // stay in START
}

int QUOTEstate(int c)
{
    if (c == '\\')
        return SLASH;       // backslash ignore next char
    else if (c == '"')
        return START;       // closing " go to START
    return QUOTE;
}

int SLASHstate()
{
    return QUOTE;
}
```


x

Programming a Deterministic Finite Automaton – noq.c

primary loop read a char at a time until EOF

process input based on current state




```
int main(void)
{
    int c;                  // input char
    int state = START;      // initial state of DFA

    while ((c = getchar()) != EOF) {
        switch (state) {
        case START:
            state = STARTstate(c);
            break;
        case QUOTE:
            state = QUOTEstate(c);
            break;
        case SLASH:
            state = SLASHstate();
            break;
        default:
            fprintf(stderr, "Error: Invalid state (%d)\n");
            return EXIT_FAILURE;
        }  // end switch
    }  // end while
    /*
     * All done. No explicit end state used here.
     * if not in start state, we have an error
     */
    if (state == START)
        return EXIT_SUCCESS;
    // ok we had an error
    fprintf(stderr, "noq error: Missing end quote \"\n");
    return EXIT_FAILURE;
}
```

call state handlers based on current state
state handlers return next state

check ending "state"

x

Aside: Remember make from CSE15L?

```
# CSE30SP24 DFA Example

# if you type 'make' without arguments, this is the default
PROG        = noq
all:        $(PROG)

# header files and the associated object files
HEAD        =  states.h
SRC         =  noq.c states.c
OBJ         =  ${SRC:%.c=%.o}

# special libraries
LIB         =
LIBFLAGS    = -L ./ $(LIB)

# select the compiler and flags you can over-ride on command line
# e.g., make DEBUG=
CC          = gcc
DEBUG       = -ggdb
CSTD        =
WARN        = -Wall -Wextra
CDEFS       =
CFLAGS      = -I. $(DEBUG) $(WARN) $(CSTD) $(CDEFS)

$(OBJ):     $(HEAD)

# specify how to compile/assemble the target
$(PROG):    $(OBJ)
    $(CC) $(CFLAGS) $(OBJ) $(LIB) -o $@

# remove binaries
.PHONY: clean clobber
clean:
    rm -f $(OBJ) $(PROG)
```


X

Programming a Deterministic Finite Automaton - testing

```
$ make
gcc -I. -ggdb -Wall -Wextra    -c -o noq.o noq.c
gcc -I. -ggdb -Wall -Wextra     -c -o states.o states.c
gcc -I. -ggdb -Wall -Wextra   noq.o states.o  -o noq
$ ./noq
123"456"789
123789
"123"45"67"
45
"123
456
78"9
9
"test
^d
noq error: Missing end quote "
$ cat in
line1"34"
"line2"line2
line3"
line4
$ ./noq < in > out
noq error: missing end quote "
$ cat out
line1
line2
line3$
```

typed input in red
output in blue

x