Exceptions

CS 217

Handling Errors in C

- A global error flag `errno` to remember the last error of the system call
- Use `perror(const char *)`to print out the meaning of the error to `stderr`
  - Example

```
#include <stdio.h>

foo(...){
   ...
   perror("In function foo");
}
```

Handling Errors in C

- Return errors from a function

```
int foo( ... );

if (foo(...) == ERROR) {
   printf("error in function foo\n");
   exit(1);
}
```

- Problems
  - Client code may not check the error codes
    - `printf` returns the number of arguments successfully printed
    - Who checks that?
  - You may not have a chance to return an error code
    - Your code may have a divide-by-zero error

A C++ Example using exceptions

```
foo(void) {
   char *buf;
   buf = new char[512];
   if( buf == 0 )
      throw "Memory allocation failure!";
   ...
}
main(void) {
   try {
      foo();
   } catch( char * str ) {
      cout << "Exception raised: "
           << str << '\n';
   }
   // ...
}
```

A C++ Example using exceptions

```
foo(void) {
   char *buf;
   buf = new char[512];
   if( buf == 0 )
      throw "Memory allocation failure!";
   ...
}

bar() {foo(); x = 0;}

main(void) {
   try {
      bar();
   } catch( char * str ) {
      cout << "Exception raised: "
           << str << '\n';
   }
   // ...
}
```

Exception Handling in Languages

- Modern languages (Modula-2, Modula-3, C++, Java, etc) provide ways to handle exceptions
  - Programs can raise an exception
  - Catch the exception and handle it
- Try-Catch-Throw in C++

```
try {
   // code to be tried
   foo();
} catch (type exception) {
   // code to be executed in case of exception
}
```

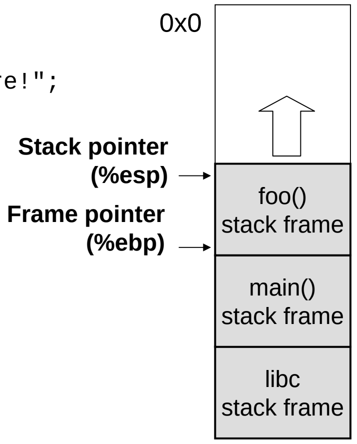
Exception Handling in Languages

- Modern languages (Modula-2, Modula-3, C++, Java, etc) provide ways to handle exceptions
  - Programs can raise an exception
  - Catch the exception and handle it
- Try-Catch-Throw in C++

```
try {
   // code to be tried
   if (flag) throw exception;
   . . .
} catch (type exception) {
   // code to be executed in case of exception
}
```

A C++ Example

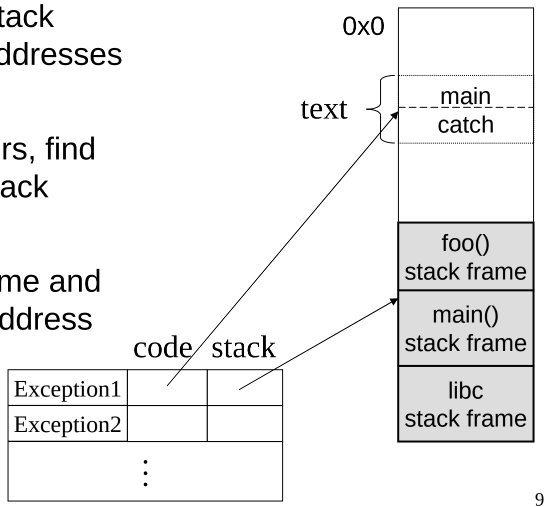
```
#include <iostream>
using namespace std;
foo(void) {
   char *buf;
   buf = new char[512];
   if( buf == 0 )
      throw "Memory allocation failure!";
   ...
}
main(void) {
try {
      foo();
   }
   catch( char * str ) {
      cout << "Exception raised: "
           << str << '\n';
   }
      // ...
}
```



How do you implement this stuff?

Implementation Consideration

- For every “try-catch-throw”, register the scope (stack frame) and “catch” addresses in a data structure
- When a “throw” occurs, find the closest “catch” stack frame
- Unwind the stack frame and jump to the “catch” address



More Implementation Considerations

- Can be implemented by a signal handler
  - For each “try-catch-throw”, register the scope (stack frame) and install a signal handler for finding the catch handler
  - When an exception occurs, OS invokes the handler which finds the closest “catch” stack frame
  - Unwind the stack frame and jump to the “catch” address
- How does a signal handler jump to the “catch” address?