## Exception Handling

- Runtime error checkings cannot be replaced by assertions. Why?
- A typical method in C is “check, print, and exit:”

```
void *malloc( size_t size ) {
    char *p;
    ...
    if (...)
        return NULL;
    else
        return p;
}
Line_T Line_New( char *text )
{
    Line_T line;

    line = malloc( sizeof *line );
    if ( line == NULL ) {
        fprintf( stderr, “Out of memory.\n” );
        exit( 0 );
    };
}
```

## Disadvantage of Runtime Checking

- **The checking is not propagated back to nested caller**

```
void Ed_Edit_Paste( void ) {
    ...
    p = Line_New( buf );
    ...
}
```

- **What we want is to save the current file if “Out of memory” occurs**
- **Can we do the following?**

```
Line_T Line_New( char *text )
{
    Line_T line;

    line = malloc( sizeof *line );
    if ( line == NULL ) {
        fprintf( stderr, “Out of memory.\n” );
        File_Save( File_name );
        exit( 0 );
    };
}
```

## Another Alternative

- Passing a status code through all calls

```
#define malloc_Okay        0xffffffff
#define malloc_OutOfMemory 0xffff0000
void * malloc( size_t size, int* status ) {
    char *p;
    ...
    *status = malloc_Okay;
    if ( ... ) {
        *status = malloc_OutOfMemory;
    return p;
}

Line_T NewLine( int x, int *status )
{
    Line_T line; int myStatus;

    p = malloc( sizeof *line, &myStatus );
    *status = myStatus;
        ...
}
```

## Another Alternative, Cont’d

- **The caller of Line_New() then becomes:**

```
void Ed_Edit_Paste( void ) {
    int status;

    ...
    p = Line_New( buf, &status );
    if ( status == malloc_OutOfMemory ) {
        File_SaveOnYes( File_name );
        exit( 0 );
    }
    ...
}
```

- **What are the pros and cons of this approach?**

## Exception Handling

- Avoid always passing status code
- Catch errors only when you want to
- Raise an exception

```
throw “Out of memory”;
```

- **Try and catch**

```
try {
    ...             /* nested calls can raise exceptions */
}
catch ( “Out of memory” ) {
    ...             /* exception handling */
}
```

## Exception Handling, Cont’d

- **Use throw to raise an exception**

```
void * malloc( size_t size ) {
    char *p;
    ...
    if ( ... )
        throw “Out of memory”;
    else
        return p;
}
```

- **No checking is necessary unless you need to catch**

```
Line_T Line_New( char *text ) {
    Line_T line;
    ...
    line = malloc( sizeof *line );
    ...
    return line;
};
```

## Exception Handling, Cont'd

- **Try and Catch**

```
Ed_Edit_Paste( void ) {
    ...
    try {
        line = Line_New( buf );
        ...                          /* no checking */
    }
    catch ( "Out of memory" ) {
        File_SaveOnYes( File_name );
        ...
        exit( 0 );
    }
```

## Exception Handling in Modula-3

- Raise an exception

```
RAISE( exception );
```

- **Catch exceptions**

```
TRY
    ...
EXCEPT
|   exception1: handler1;
|   exception2: handler2;
    ...
|   ELSE    handlerN;
END;
```

- **Finalization**

```
TRY
    ...
FINALLY
    ...
END;
```