High Tech Disasters

You may be the comic relief!
Disaster #1: All Stars
Disaster #1 – “All Stars”

- “All Stars” may not be team players
  - Everyone had to have input on every decision
  - Everyone wanted to run their own world
  - “We did it this way at XYZ”
  - Internal competition may kill the enterprise

- Inexperience is a bad teacher
  - I didn’t know what was good or bad
  - Smart people do stupid things for good reasons
Disaster #2: Stupidity
Lessons Learned

- You can’t manage something you don’t understand (at least a little).
- Attitude is more important than knowledge.
- In the land of the blind, the one eyed man is considered insane.
- Specialists aren’t generalists.
- Specialists in different areas probably can’t communicate with each other well.
- Many people need to repeat 3rd grade.
- Sometimes ‘many people’ means you.
Disaster #3: Internal Competition
Server Debug Cycle

Result: 2 Turnarounds per day
Workstation Debug Cycle

Result: 4 Turnarounds per hour
Results of Debug Disparity

- **Workstation**
  - Shipped on time
  - I offered to share resources & techniques

- **Server manager: “That’s not fair.”**
  - His veteran engineers couldn’t program
  - Tried to take away my workstations
  - Server schedule slipped by months
Disaster #4: Self Inflicted Wounds
Another useful form of automatic type conversion is that relational expressions like \( i > j \) and logical expressions connected by && and || are defined to have value 1 if true, and 0 if false. Thus the assignment

\[
\text{isdigit} = c \geq '0' \&\& c \leq '9';
\]

sets \text{isdigit} to 1 if \( c \) is a digit, and to 0 if not. (In the test part of if, while, for, etc., “true” just means “non-zero.”)

Implicit arithmetic conversions work much as expected. In general, if an operator like + or * which takes two operands (a “binary operator”) has operands of different types, the “lower” type is promoted to the “higher” type before the operation proceeds. The result is of the higher type. More precisely, for each arithmetic operator, the following sequence of conversion rules is applied:

converted to the type of the left, which is the type of the result. A character is converted to an integer, either by sign extension or not, as described above. The reverse operation, int to char, is well-behaved — excess high-order bits are simply discarded. Thus in

\[
\text{int } i;
\text{char } c;
\]

\[
i = c;
\text{c } = i;
\]

the value of \( c \) is unchanged. This is true whether or not sign extension is involved.

If \( x \) is float and \( i \) is int, then

\[
x = i
\]

and

Otherwise the operands must be int, and the result is int.

Notice that all float’s in an expression are converted to double; all floating point arithmetic in C is done in double precision.

Conversions take place across assignments; the value of the right side is converted to the type of the left, which is the type of the result. A char
Lessons Learned

- There is no substitute for domain-specific expertise.
- Don’t let your competitor tell you something about your own product you don’t know.
- Know when to cut your losses.
STOP
IN THE
NAME OF
HUMANITY