Fundamentals of MCS (CS)

Overview of Part II

2014.11.17, Ken Wakita

Kamp, *The most expensive one-byte mistake*, ACM Queue, 9(7), 2011.

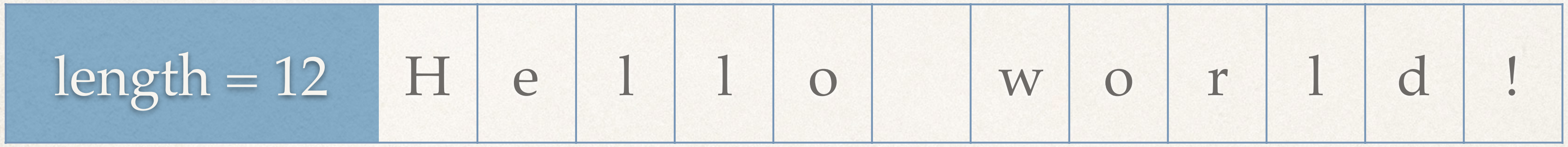
- Q: Which IT or CS decision has resulted in the most expensive mistake?
    - Sony's troubles with PlayStation Network (2011)
    - IBM's choice of B. Gates (MS-DOS) over G. Kildall (CP/M), (1976)
    - Fortran syntax anomaly

DO bug spotted in mission control system for Mercury

- **DO** statement (similar to C's **for** statement)
    - **DO** 5 K = 1,3
      things to repeat (for K = {1, 2, 3}, repeat commands upto the line labeled 5)
    - **DO** 5 K = 1.3
- Q: Can you identify the reason why the second program executes "things to repeat" only once?
    - Fortran compiler's interpretation
        - DO5K = 1.3

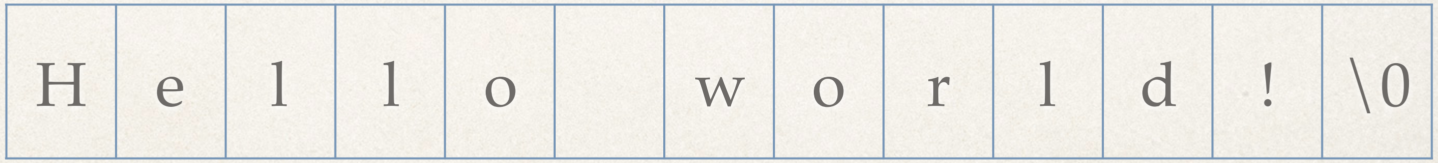
String formats

- Address + length (Fortran, Pascal, Java, ML, …)



  - 4 bytes overhead on 32bits memory address space

- Address + magic marker such as ‘\0’ (C)



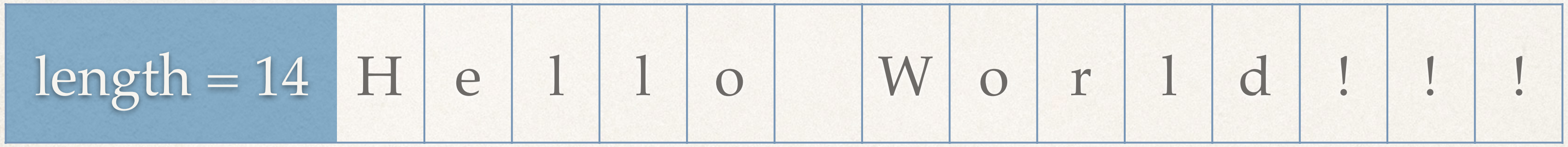
  - 1 byte overhead

Costs of '\0' decision

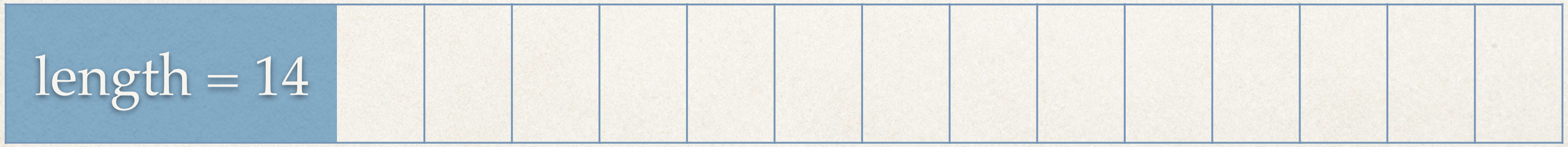
- Hardware: addition of **logical string assist** instructions to CPU instruction set
- bcopy/memcopy inefficiency
- Growing complexity of compiler construction

Making a string copy (address+len)

- Step 1

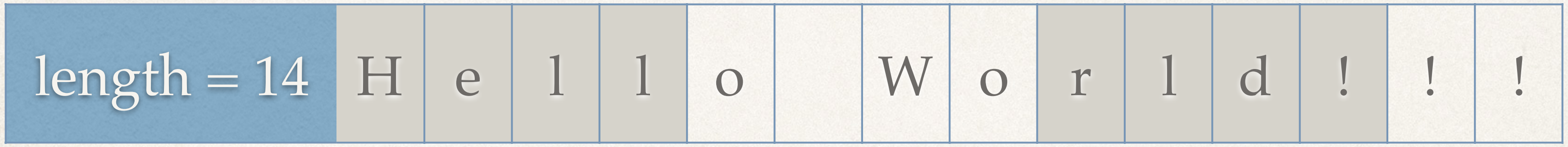


- Identify the length of the original string and allocate as memory chunk of required size, which is (4 + len) bytes. Then fill the length field of the new area.

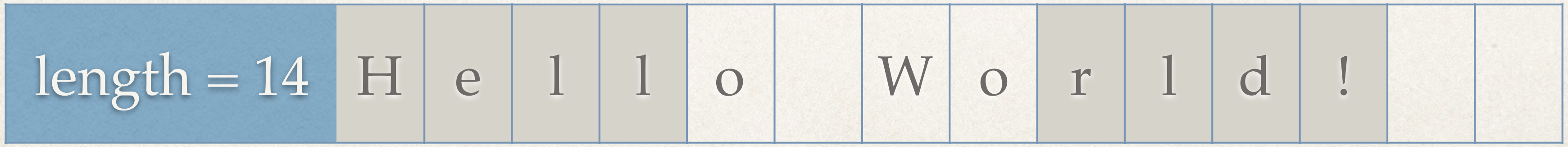


Making a string copy (address+len)

- Step 2

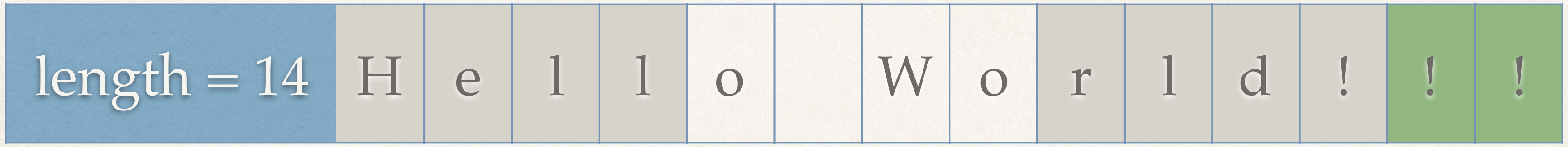


- Copy the first three worlds, using the **word-copy** instruction. (three copies)

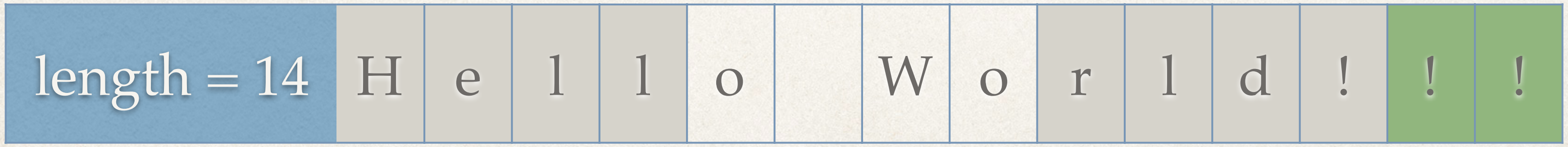


Making a string copy (address+len)

- Step 3



- Copy the remaining two letters, using the **byte-copy** instruction. (two copies)
  Three word copies + two byte copies in total.



Making a string copy (address+magic)

- Step 1

H	e	l	l	o		W	o	r	l	d	!	!	!	\0

- Identify the length of the original. To do this, we need to scan the whole string from left to right unto '\0' counting the number of letters. At least four word-aligned memory accesses and 15 times of comparison.

Making a string copy (address+magic)

- Step 2

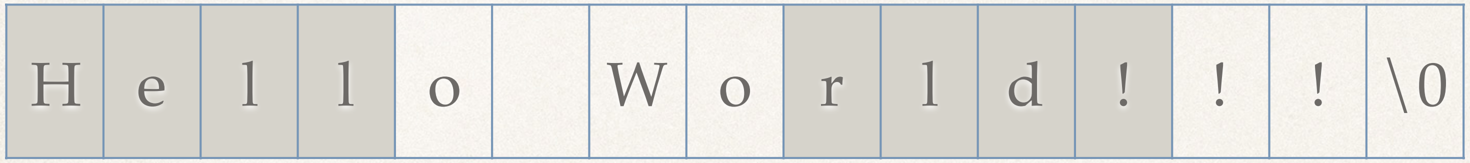
H	e	l	l	o		W	o	r	l	d	!	!	!	\0

- Allocate a chunk of memory and place a magic code ('\0') at its end. (one memory access)

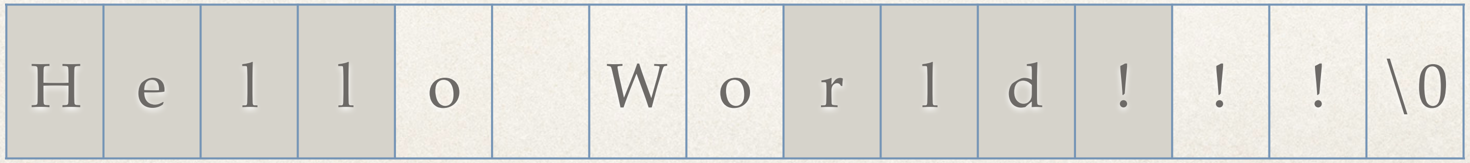
														\0

Making a string copy (address+magic)

- Step 3



- Copy the content of the string in a similar fashion as we did for the address+length format. As a whole address+magic requires 4 more memory accesses and 15 more comparisons.



Smart Compiler: strcpy → memcpy

- strcpy(from, to) vs. memcpy(from, to, length)
- If the address-length string format is used, the compiler can simply replace usages of strcpy with faster memcpy.
- If the address-magic string format is used, implementation of the above technique requires the compiler to identify the string size at compiler time.

Security threat

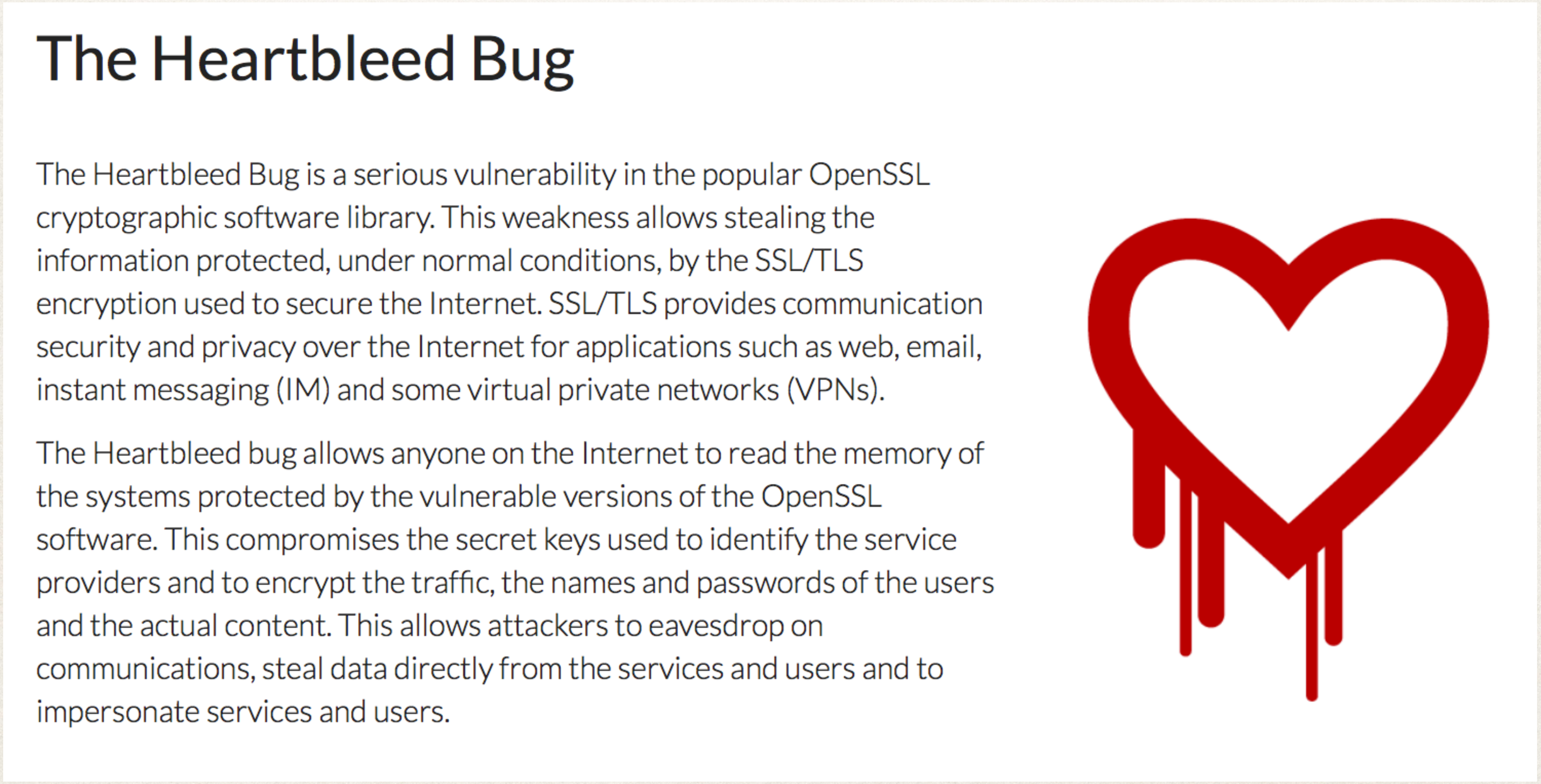
- Stack smashing attack (or buffer overflow/overrun)
    - Aleph One, Smashing the stack for fun and profit, BugTraq 7(49), 1996.

Security Cost

Mitigation of these risks has been added at all levels. Long-missed no-execute bits have been added to CPUs' memory management hardware; operating systems and compilers have added address-space randomization, often at high costs in performance; and static and dynamic analyses of programs have soaked up countless hours, trying to find out if the byzantine diagnostics were real bugs or clever programming.

- Memory Protection (H/W, OS, Compiler)
- Address-space randomization
- Static/dynamic analysis

HeartBleed Bug (Apr. 2014)



The Heartbleed Bug

The Heartbleed Bug is a serious vulnerability in the popular OpenSSL cryptographic software library. This weakness allows stealing the information protected, under normal conditions, by the SSL/TLS encryption used to secure the Internet. SSL/TLS provides communication security and privacy over the Internet for applications such as web, email, instant messaging (IM) and some virtual private networks (VPNs).

The Heartbleed bug allows anyone on the Internet to read the memory of the systems protected by the vulnerable versions of the OpenSSL software. This compromises the secret keys used to identify the service providers and to encrypt the traffic, the names and passwords of the users and the actual content. This allows attackers to eavesdrop on communications, steal data directly from the services and users and to impersonate services and users.

Theme of part II:
Technologies for development of reliable software

The rest of part II

- Nov. 26 (Wed): Type System: Why Types Matters?
- Dec. 1: Grammar, Semantics, Type Relation
- Dec. 8: Mechanical Proof Assistance System (1/2)
- Dec. 15: Mechanical Proof Assistance System (2/2)

Information on Part II talks

- Course website (teaching materials are here):
  - https://github.com/wakita/fmcs2015
- Questions and Comments should be addressed by creating a **New issue** at GitHub.
- Other notes: my public Evernote notebook. The URL is offered on the course website.