

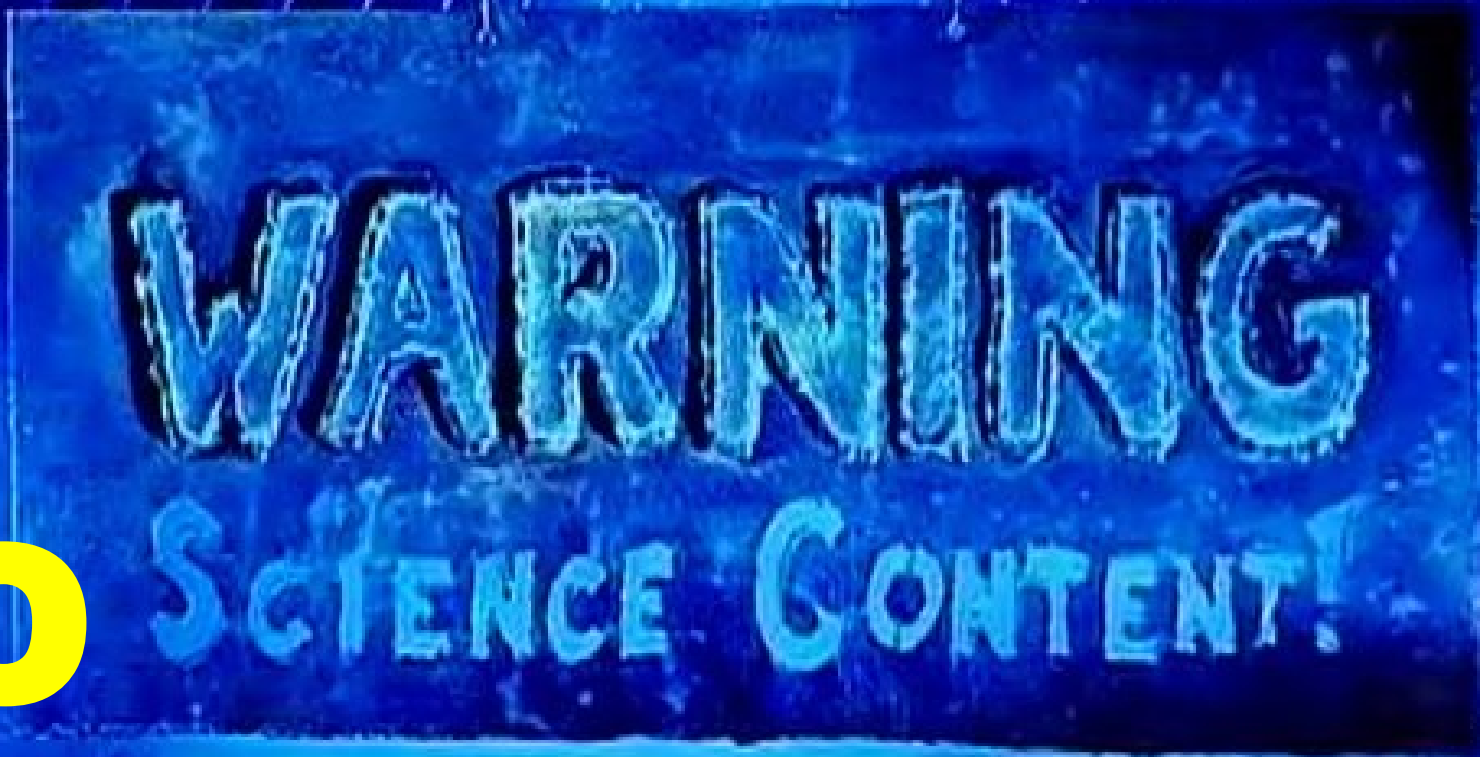
# Reconstructive Software Archaeology

**Warren Toomey**  
**School of IT, Bond University**

This is a case study in restoring the  
1<sup>st</sup> Edition of UNIX from 1971.

The restoration is interesting in itself, but it  
also raises issues that are relevant to  
other software fields.

**NO**



Happy 40<sup>th</sup> Birthday, UNIX!



# Issues in Restoring A Computing Artifact

- Computing artifact: hardware, software
- Other resources: documentation, blueprints, schematics, configuration files, notes, written and oral anecdotes, contemporary publications
- What issues need to be considered when restoring a computing artifact to working order?

What if the artifact's purpose is unknown?



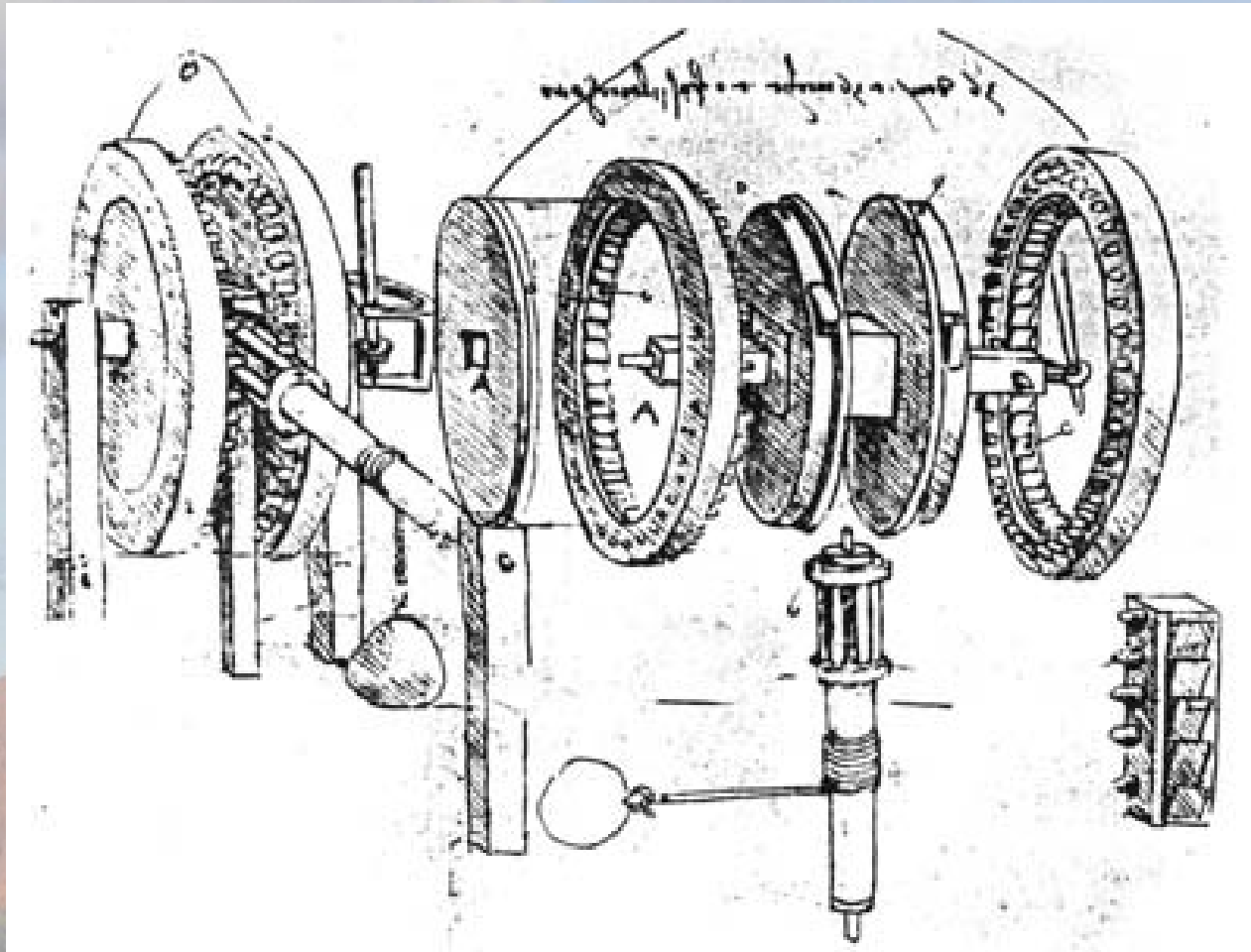
What if the documentation  
is missing?



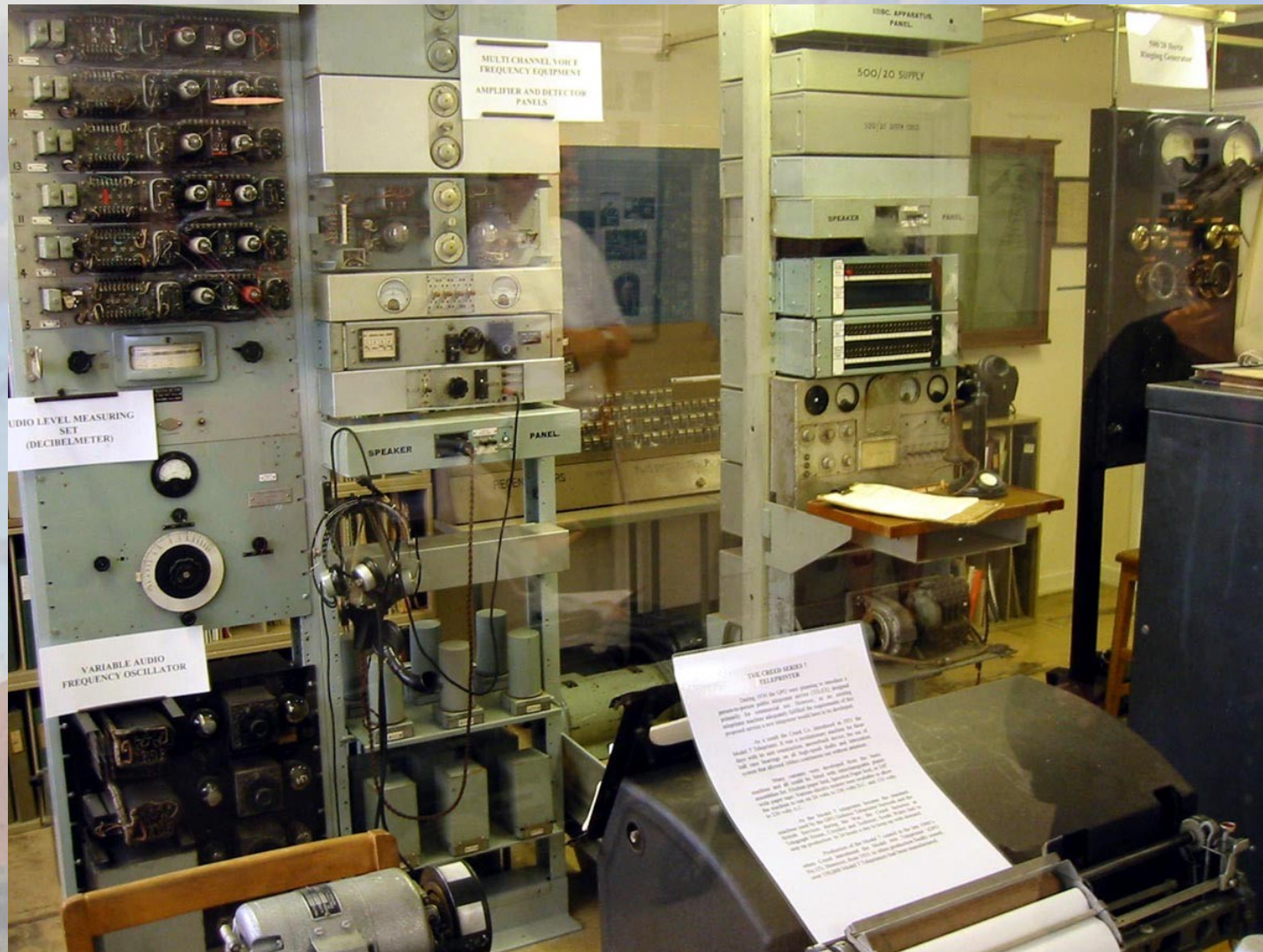
# What if the documentation is incomplete?



Is the artifact a blueprint?  
Can it be rebuilt?



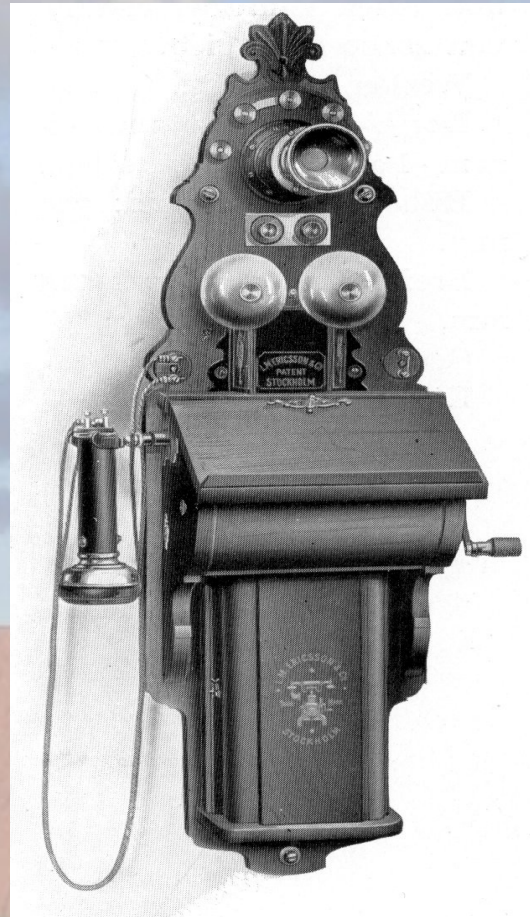
# Do we have the tools to rebuild it?



Do we have to replace some of the parts of the artifact?



Do we have to make significant changes to make it work?



# Software Restoration Issues

- Unlike physical hardware, software does not decay (at least, not while pristine copies exist)
- But in practice, software tends to exhibit what is commonly known as “*bit rot*”
- If software does not decay, then what causes the bit rot?
- Bit rot is a function of the software's *environment*, and not the software itself

# The UNIX Heritage Society

- I'm a founding member of the Unix Heritage Society. Our aim is to preserve the knowledge and artifacts of early UNIX
- Where possible, we try to keep old systems working. Past successes:
  - Restoration of earliest C version of UNIX: 1973
  - Restoration of earliest C compiler: also 1973
  - Creation of executable environment for UNIX user-mode binaries, assembled in 1972
- The 1<sup>st</sup> Edition of UNIX, from 1971, was lost

# 1<sup>st</sup> Edition UNIX Features

- Hierarchical filesystem: files, directories, subdirectories
- Pre-emptive multitasking & processes
- A flexible command-line interpreter
- Multiuser, including e-mail
- Mountable storage making a single filesystem tree
- Hard links: a file can have multiple names
- Multiple languages: assembly, FORTRAN, Basic, TMG, shell scripting

# 1<sup>st</sup> Edition UNIX



Dennis & Ken at the PDP-11/20 console

# And then...

- A paper document containing a listing of the 1<sup>st</sup> Edition UNIX kernel was found

```

                                UNIX IMPLEMENTATION

/ u1 -- unix

unkni: / used for all system calls
sysent:
    incb    sysflg / indicate a system routine is
    beq     1f / in progress
    jmp     panic / called if trap inside system

1:
    mov     $$syst+2,clockp
    mov     r0,-(sp) / save user registers
    mov     sp,u.r0 / pointer to bottom of users stack in u
    mov     r1,-(sp)
    mov     r2,-(sp)
    mov     r3,-(sp)
    mov     r4,-(sp)
    mov     r5,-(sp)
    mov     ac,-(sp) / "accumulator" register for extended
                    / arithmetic unit
    mov     mq,-(sp) / "multiplier quotient" register for t
                    / extended arithmetic unit
    mov     sc,-(sp) / "step count" register for the extend
                    / arithmetic unit
    mov     sp,u.sp / u.sp points to top of users stack
    mov     18.(sp),r0 / store pc in r0
    mov     -(r0),r0 / sys inst in r0      10400xxx
    sub     $$sys,r0 / get xxx code
    asl    r0 / multiply by 2 to jump indirect in bytes
    cmp     r0,$2f-1f / limit of table (35) exceeded
```

# Can It Be Restored?

- Needs to be OCR'd and eyeballed
- Contradictory typed & handwritten comments
- No 1<sup>st</sup> Edition assembler, only later ones
- No bootstrap code in any form
- No filesystem or creation tool, just the docs
- Need a PDP-11/20 simulator: one exists, but not all the required hardware
- Not sure if existing executables are from 1<sup>st</sup> Edition or 2<sup>nd</sup> Ed: will they be compatible?

# What was Done, Part 1

- Document scanned, OCR'd, manually checked & cross-checked by ~10 people
- Tool written to modify output from 7<sup>th</sup> Edition assembler to be compatible with 1<sup>st</sup> Edition assembler
- Existing Apout tool allows 7<sup>th</sup> Ed assembler to run without a full PDP-11 simulator
- Several logic errors and missing lines found in the paper listing: fixed
- KE11A support added to PDP-11 simulator
- Result: kernel runs to a point, then hangs

# What was Done, Part 2

- “Cold” kernel fixed, builds near-empty filesystem. Result: “Warm” kernel hangs
- *mkfs* tool written to build and fully populate the root and /usr filesystems
- Result: UNIX boots, *init*, login & shell work!
- Simulator further modified to emulate DC-11
- Result: multiuser UNIX system
- Kernel modified to deal with “0407” executables
- Result: all old executables run; C compiler runs and can recompile itself

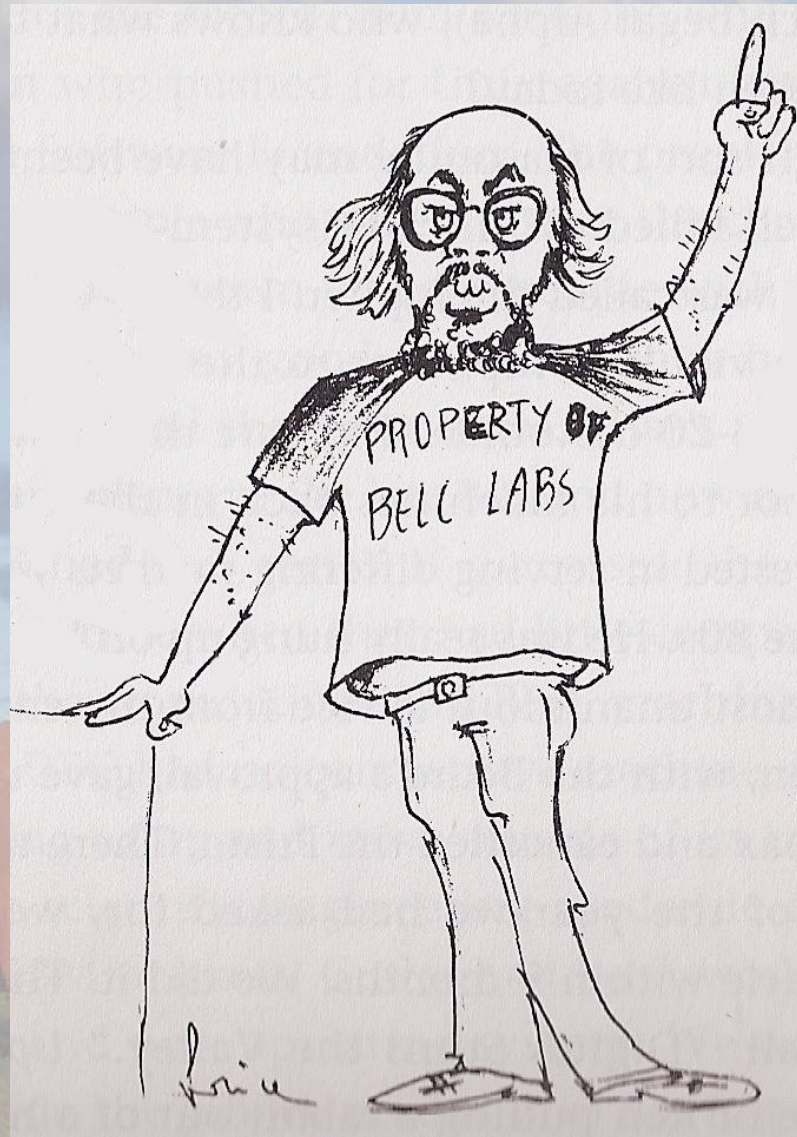
# Software Reconstruction

- Software suffers from “bit rot”. We had to:
  - Fix typos, missing lines, logic mistakes in the source code
  - Build tools which could assemble the source code, and construct suitable filesystems
  - Modify an existing PDP-11 simulator to provide an executable environment for the system
  - Interpret old documentation: on the whole, it was excellent, but it was vague or omitted details in places
- Luck played a role: documentation, preserved executables, existing tools

# Lessons Learned for Now

- Write good documentation
- Keep software current on new platforms
- If necessary, write simulators now while the hardware details still exist
  - Moore's Law helps here
- All software requires an environment. Take a crucial component away & it stops working:
  - Hardware, compilation tools, user manual, filesystem, even configuration files
- As system complexity increases, the work needed to resurrect/restore increases

# Questions?



# Old & New System Calls

<b>1<sup>st</sup> Edition</b>	<b>Linux 2.6</b>	<b>1<sup>st</sup> Edition</b>	<b>Linux 2.6</b>
1: exit	exit	15: chmod	chmod
2: fork	fork	16: chown	lchown
3: read	read	17: break	unused
4: write	write	18: stat	stat
5: open	open	19: seek	lseek
7: wait	waitpid	20: tell	getpid
8: creat	creat	21: mount	mount
9: link	link	22: umount	umount
10: unlink	unlink	23: setuid	setuid
11: exec	execve	24: getuid	getuid
12: chdir	chdir	25: stime	stime
13: time	time	26: quit	ptrace
14: mkdir	mknod	28: fstat	fstat