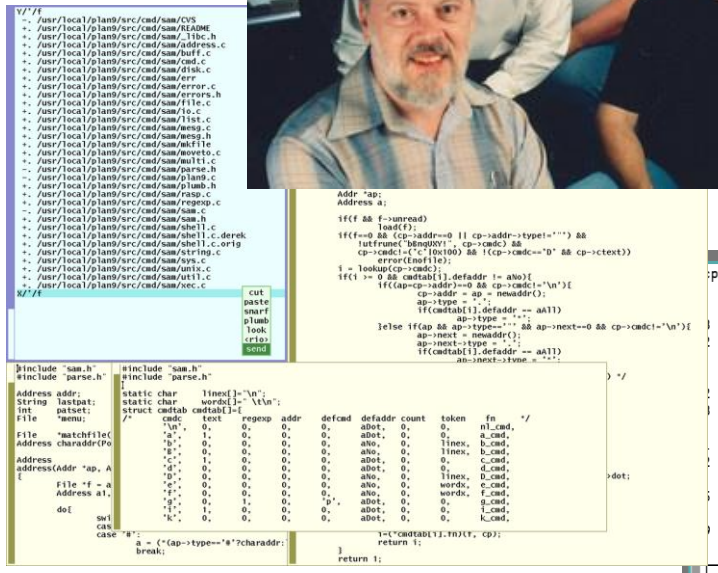
Plan 9

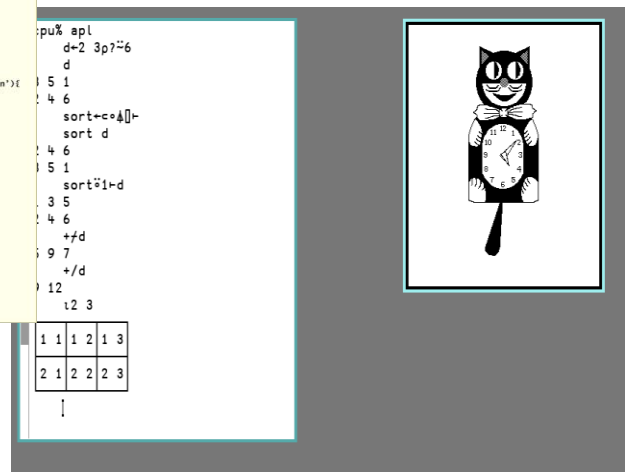
A New Paradigm for Operating Systems

```
term% fortune
Your fantasy will come true.
```

A Brief History

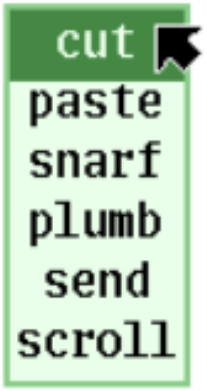
- Began development in late 80s
- Development led by a star-studded cast
  - Rob Pike, Brian Kernighan, Ken Thompson, Bjarne Stroustrup
- Released to public in 1995
- Fine-tuned for research and networking – not user-friendly
- Unix: “Everything is a file”
  - Minimalism, portability, purity
- Laid to rest Jan 10, 2015
- Carried on by 9front, Inferno, 9legacy, etc.

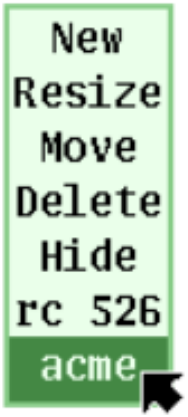




Idiosyncrasies

- Independent C library
  - No GNU, and no GCC! A dialect of C.
- Custom, but portable, C compilers
  - Unique compiler for each architecture
  - 8c for x86, 6c for AMD, 5c for ARM, 0c for MIPS
- **9P** network protocol
  - Used by Microsoft for WSL since 2019!
- Independent windowing system (**rio**)
  - Doesn't use X Window System
- A distributed operating system
  - Total data communication transparency
  - Encourages cluster computing
- Not Bash, not PowerShell, but **rc**!
  - Some commands are the same, many are different
  - No symbolic links, and no sudo (it's unsafe!)






```
term% lc
bin      hello.c lib      tmp
term% 8c hello.c
term% 8l hello.8
term% lc
8.out    hello.8 lib
bin      hello.c tmp
term% 8.out
hello world
term%
```

Synchronization via Channels

- Threads communicate via **channels**
- Mutexes, condition variables, and locks are not necessary
  - Although they are provided
- **send()**: sends value through channel, blocks
- **recv()**: receives value through channel, blocks
- **chancreate()/chanclose()** enforce synchronization – no primitives needed
- Channels are powerful enough to implement **pipes**, **sockets**, and **semaphores**

```
term% lc
acme.dump   lib         readme.acme tmp
bin         proj        readme.rio
term% cd proj
term% ls
8.out
proj.8
proj.c
term% 8c proj.c
term% 8l proj.8
term% ./8.out
started lockstep up to 20 iterations
initiator: 1
     responder: 2
initiator: 3
     responder: 4
initiator: 5
     responder: 6
initiator: 7
     responder: 8
initiator: 9
     responder: 10
initiator: 11
     responder: 12
initiator: 13
     responder: 14
initiator: 15
     responder: 16
initiator: 17
     responder: 18
initiator: 19
     responder: 20
initiator detected max count (20), closing channel
     responder: channel closed: exiting.
term%
```

```
Newcol Kill Putall Dump Exit
New Cut Paste Snarf Sort Zerox Delcol
proj.c Del Snarf Undo | Look
// demonstrates channels

#include <u.h>
#include <libc.h>
#include <thread.h>

#define MAX_VALUE 20

Channel* initiator_channel;
Channel* responder_channel;

void
initiatorthread(void*)
{
   int control;

   for(;;) {
      recv(initiator_channel, &control);

      if (control >= MAX_VALUE) {
         print("initiator detected max count (%d), closing channel
\n", control);
         chanclose(responder_channel);
         threadexits(nil);
      }

      control++;
      print("initiator: %d\n", control);
      send(responder_channel, &control);
   }
}

void
responderthread(void*)
{
   int control;

   for(;;) {
      if(recv(responder_channel, &control) < 0) {
```

Concurrency in Plan 9

- Same capabilities as POSIX threads, but with a different library
- The iconic producer-consumer problem
- Again, we use channels to avoid race conditions
- Channels are “typed”, unlike Linux
  - Recall: **send()** vs. **sendul()**, and **recv()** vs. **recvul()**
  - Untyped (byte stream, pointers) vs. typed (direct, pass by value)
- Unlike Linux, no thread attributes, no joining
- Writers and readers automatically synchronize thanks to rendezvous communication

```
term% lc
acme.dump   lib
bin         proj
term% cd proj
term% 8c proj.c
term% 8l proj.8
term% ./8.out
p 1
p 2
        c 1
p 3
        c 2
p 4
        c 3
p 5
        c 4
p 6
        c 5
p 7
        c 6
p 8
        c 7
p 9
        c 8
p 10
        c 9
p 11
        c 10
p 12
        c 11
p 13
        c 12
p 14
```

Newcol Kill Putall Dump Exit
New Cut Paste Snarf Sort Zerox Delcol
+Errors Del Snarf | Look
Del Snarf Undo | Look

```
#include <u.h>
#include <libc.h>
#include <thread.h>

#define MAX_VALUE 20

Channel* initiator_channel;
Channel* responder_channel;

void
initiatorthread(void*)
{
   int control;

   for(;;) {
      recv(initiator_channel, &control);

      if (control >= MAX_VALUE) {
         print("initiator detected max count (%d), closing channel\n", control);
         chanclose(responder_channel);
         threadexits(nil);
      }

      control++;
      print("initiator: %d\n", control);
      send(responder_channel, &control);
   }
}
```

/usr/glenda/proj/proj.c Del Snarf Get | Look

Newcol Kill Putall Dump Exit
New Cut Paste Snarf Sort Zerox Delcol
/usr/glenda/ Del Snarf Get | Look

```
acme.dump    lib/          readme.rio
bin/         readme.acme   tmp/
```

Del Snarf Undo | Look

```
void
responderthread(void*)
{
   int control;

   for(;;) {
      if(recv(responder_channel, &control) < 0) {
         print("\t responder: channel closed: exiting.\n");
         threadexits(nil);
      }

      control++;
      print("\t responder: %d\n", control);
      send(initiator_channel, &control);
   }
}

void
threadmain(int, char*[])
{
   int start;

   initiator_channel = chancreate(sizeof(int), 0);
   responder_channel = chancreate(sizeof(int), 0);

   threadcreate(initiatorthread, nil, 8*1024);
   threadcreate(responderthread, nil, 8*1024);

   start = 0;
   print("started lockstep up to %d iterations\n", MAX_VALUE);
   send(initiator_channel, &start);

   threadexits(nil);
}
```

System Calls

- Note the presence of a kernel-side **note system**
  - *Pun not intended*
- Single words of data may be exchanged via **rendezvous**
- Plan 9: **38 syscalls**; Linux: Over 400
- Note: Plan 9 has **no kernel threads**!

Memory management	
brk_	allocate memory
segattach	attach to or create a segment
segdetach	detach from a segment
segfree	free physical memory
segbrk	change segment length
segflush	flush cache

Notes	
alarm	set the alarm timer
notify	set the note handler
noted	continue after note

Files	
open	open an existing file
create	create a new file or open an existing file for writing
pread	read from an open file
pwrite	write to an open file
chdir	change current directory
seek	change the current position in an open file
close	close a file descriptor
dup	duplicate a file descriptor
fd2path	retrieve file name
stat	read file metadata
fstat	read open file metadata
wstat	write file metadata
fwstat	write open file metadata
remove	delete a file

Namespace management	
mount	mount a 9P connection
bind	bind a file or directory
unmount	unmount or remove a bind
9P connections	
fversion	initialize a 9P connection
fauth	initiate authentication

Process management	
rfork	change process attributes or create a new process
exec	replace the current process
exits	terminate the current process
errstr	exchange error string
sleep	sleep a given amount of time
Synchronization	
await	wait for a child process to terminate
pipe	create a pipe
rendezvous	exchange a word of data
semacquire	acquire a semaphore
semrelease	release a semaphore

Significance and Impact

- Once-native Plan 9 development tools, like rc, acme, and plumbing have been ported to Linux, Mac, and Windows
- The 9P filesystem is made use of today by Windows' WSL
- Served as a catalyst for the software minimalism movement
- A pioneering system in grid computing
- Served as the backbone for years of innovative research at Bell Labs
- Ran by Coraid hardware (data storage)
- Influenced Google's Go-lang
- Inspired Inferno, 9front, and other OSes