Optimizing synchronization primitives in Wine

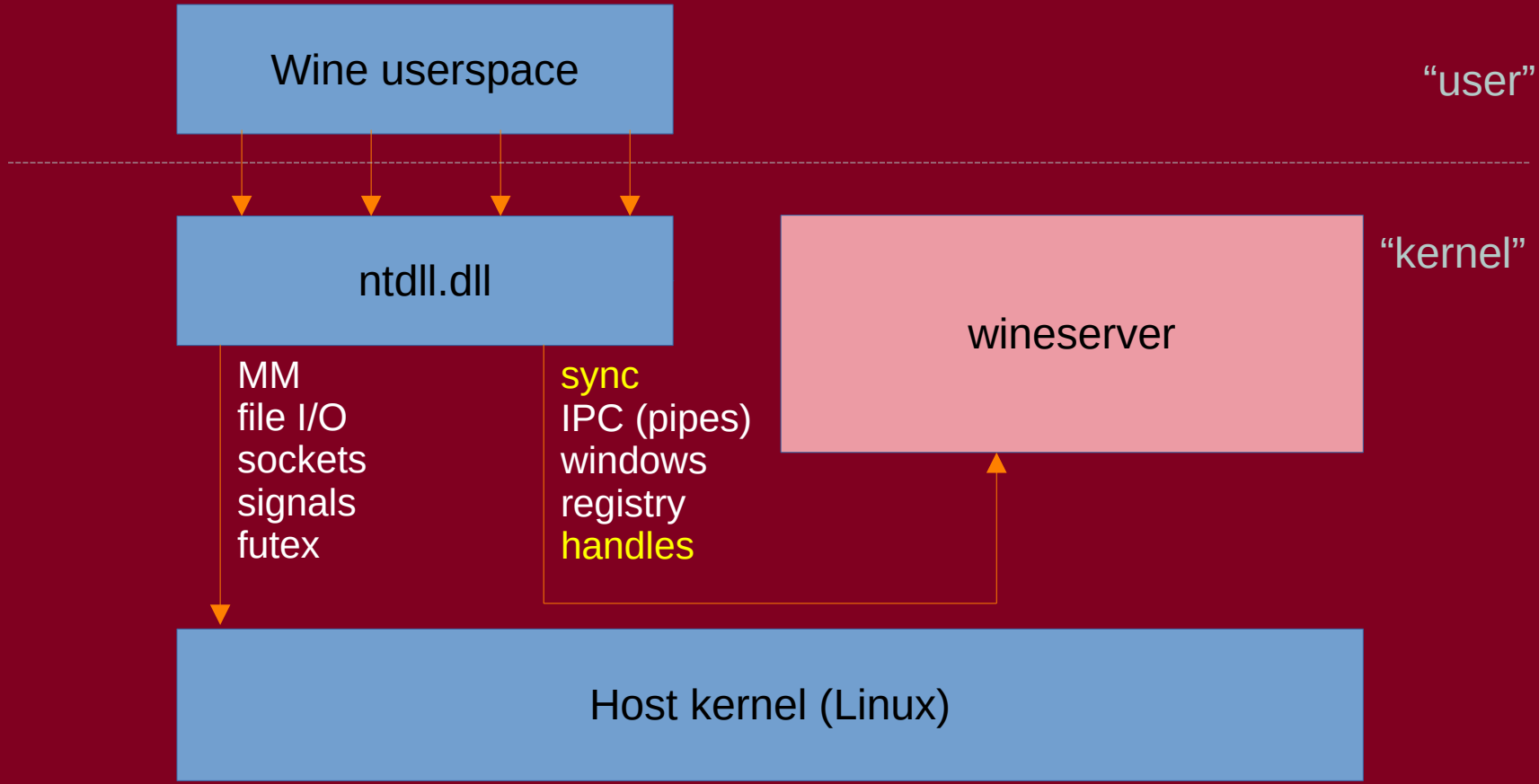
Zeb Figura

Linux Plumbers Conference 2023

About me

- Wine developer for ~6 years
- Wine work on multimedia, 3D, Win16, sockets, hardware, kernel…
- One kernel patch
- Employed by CodeWeavers
  - We do consulting and porting using Wine

Wine’s “kernel”




GPUVis
beatsaber.dat (128.97 MB)
File Options
Event Graph
8263.2434 ms
Length: 17.1052 ms
Zoom In
Zoom Out
8280.3486 ms
0) gfx
37 events
Beat Saber.exe-3173 (Beat Saber.exe-3123)
Beat Saber.exe-3173 (Beat Saber.exe-3123)
1) gfx hw
20 events
Beat Saber.exe-3173
(Beat Saber.exe-3123)
2) cpu graph
3375 events
Beat Saber.exe
Beat Saber.exe
vrdashboard
Beat Saber.exe
RenderThread
wineserver
wineserver
wineserver
wineserver

What are handles?

What are handles?

- A handle is a file descriptor
  - But need not be a file
- Handles *can* have names
  - `\??\some\path`
- Handles have access flags
  - Objects can have ACLs
- Handles are per-process
  - But they can be shared across processes

- Many different types:
  - Event
  - Mutex
  - Semaphore
  - Timer
  - Thread
  - Process
  - File
  - Pipe
  - Socket
  - Window message queue

The easy parts

- Events: `NtSetEvent()`, `NtResetEvent()`
- Mutexes: `NtReleaseMutant()`
- Semaphores: `NtReleaseSemaphore()`
- `NtWaitForMultipleObjects()`
  - It’s like `poll(2)`
  - ...but it consumes state

The hard parts

- Events: `NtPulseEvent()`
- Mutexes: abandonment (think `EOWNERDEAD`)
- Semaphores: ☺
- `NtWaitForMultipleObjects()`
  - It can wait on “all”
  - It can wait on an “alert” (Windows version of `SIGIO`)

Is there a user space solution?

esync

- Each object is backed by an `eventfd(2)`
- Extra state in shared memory
- Vectored wait via `poll(2)`

Easy	Hard
✓ `NtSetEvent()`	✗ `NtPulseEvent()`
✓ `NtResetEvent()`	✓ Abandoned mutexes
✓ `NtReleaseMutant()`	✗ Wait-for-all
✓ `NtReleaseSemaphore()`	✓ Alertable wait
✓ `NtWaitForMultipleObjects()`	

fsync

- Each object is backed by a futex word in shared memory
- Extra state in shared memory
- Vectored wait via `futex_waitv(2)`

Easy	Hard
✓ `NtSetEvent()`	✗ `NtPulseEvent()`
✓ `NtResetEvent()`	✓ Abandoned mutexes
✓ `NtReleaseMutant()`	✗ Wait-for-all
✓ `NtReleaseSemaphore()`	✓ Alertable wait
✓ `NtWaitForMultipleObjects()`	

```
1137  static long ntsync_char_ioctl(struct file *file, unsigned int cmd,
1138  »       »       »             unsigned long parm)
1139  {
1140  »       struct ntsync_device *dev = file->private_data;
1141  »       void __user *argp = (void __user *)parm;
1142
1143  »       switch (cmd) {
1144  »       case NTSYNC_IOC_CREATE_EVENT:
1145  »       »       return ntsync_create_event(dev, argp);
1146  »       case NTSYNC_IOC_CREATE_MUTEX:
1147  »       »       return ntsync_create_mutex(dev, argp);
1148  »       case NTSYNC_IOC_CREATE_SEM:
1149  »       »       return ntsync_create_sem(dev, argp);
1150  »       case NTSYNC_IOC_DELETE:
1151  »       »       return ntsync_delete(dev, argp);
1152  »       case NTSYNC_IOC_KILL_OWNER:
1153  »       »       return ntsync_kill_owner(dev, argp);
1154  »       case NTSYNC_IOC_PULSE_EVENT:
1155  »       »       return ntsync_set_event(dev, argp, true);
1156  »       case NTSYNC_IOC_PUT_MUTEX:
1157  »       »       return ntsync_put_mutex(dev, argp);
1158  »       case NTSYNC_IOC_PUT_SEM:
1159  »       »       return ntsync_put_sem(dev, argp);
1160  »       case NTSYNC_IOC_READ_EVENT:
1161  »       »       return ntsync_read_event(dev, argp);
1162  »       case NTSYNC_IOC_READ_MUTEX:
1163  »       »       return ntsync_read_mutex(dev, argp);
1164  »       case NTSYNC_IOC_READ_SEM:
1165  »       »       return ntsync_read_sem(dev, argp);
1166  »       case NTSYNC_IOC_RESET_EVENT:
1167  »       »       return ntsync_reset_event(dev, argp);
1168  »       case NTSYNC_IOC_SET_EVENT:
1169  »       »       return ntsync_set_event(dev, argp, false);
1170  »       case NTSYNC_IOC_WAIT_ALL:
1171  »       »       return ntsync_wait_all(dev, argp);
1172  »       case NTSYNC_IOC_WAIT_ANY:
1173  »       »       return ntsync_wait_any(dev, argp);
1174  »       default:
1175  »       »       return -ENOSYS;
1176  »       }
1177  }
```

The hard parts - revisited

- Events: `NtPulseEvent()`
- Mutexes: abandonment (think `EOWNERDEAD`)
- Semaphores: ☺
- `NtWaitForMultipleObjects()`
  - It can wait on “all”
  - It can wait on an “alert” (Windows version of `SIGIO`)

Handles revisited

- A handle is a file descriptor
  - But need not be a file
- Handles *can* have names
  - `\??\some\path`
- Handles can be dup'd
  - `NtDuplicateObject`
- Handles have access flags
  - Objects can have ACLs

- Many different types:
  - Event
  - Mutex
  - Semaphore
  - Timer
  - Thread
  - Process
  - File
  - Pipe
  - Socket
  - Window message queue


i4790k 70% 63°C
RAM 8.0GiB
1080Ti 74% 62°C
VRAM 4.8GiB
DXVK 109FPS 9.2ms
Frametime min: 3.7ms, max: 9321.0ms
server-side
i4790k 67% 61°C
RAM 7.9GiB
1080Ti 99% 62°C
VRAM 4.8GiB
DXVK 157FPS 6.4ms
Frametime min: 4.0ms, max: 16.2ms
eventfd
i4790k 71% 60°C
RAM 8.0GiB
1080Ti 98% 62°C
VRAM 4.8GiB
DXVK 158FPS 6.3ms
Frametime min: 3.6ms, max: 18.0ms
futex_waitv
00:01
FPS: 153
i4790k 75% 61°C
RAM 8.0GiB
1080Ti 98% 58°C
VRAM 4.3GiB
DXVK 157FPS 6.4ms
Frametime min: 2.6ms, max: 20.9ms
NTsync
(testing version)

More benchmarks

	server	ntsync	improvement
Call of Juarez	99.8	224.1	124.55%
Dirt 3	110.6	860.7	678.21%
Forza Horizon 5	108	160	48.15%
Total War Saga: Troy	109	146	33.94%
Metro 2033	164.4	199.2	21.17%
The Crew	26	51	96.15%
Resident Evil 2	26	77	196.15%
Anger Foot demo	69	99	43.48%
Lara Croft and the Temple of Osiris	141	326	131.21%
Tiny Tina’s Wonderlands	130	360	176.92%

Thanks to Dmitry Skvortsov, FuzzyQuills, OnMars

Solutions?

- Submit the driver upstream?
  - May be the best way to match Windows performance...
- Extend existing APIs?
  - But NT is ugly...
- Half-baked idea: fast user-space RPC?
  - With a single context switch?
    - This could have interesting use cases elsewhere in Wine...
- Something else?

More information

- https://www.winehq.org/
- https://repo.or.cz/linux/zf.git/shortlog/refs/heads/ntsync4
- https://repo.or.cz/wine/zf.git/shortlog/refs/heads/ntsync4
- zfigura@codeweavers.com

codeweavers