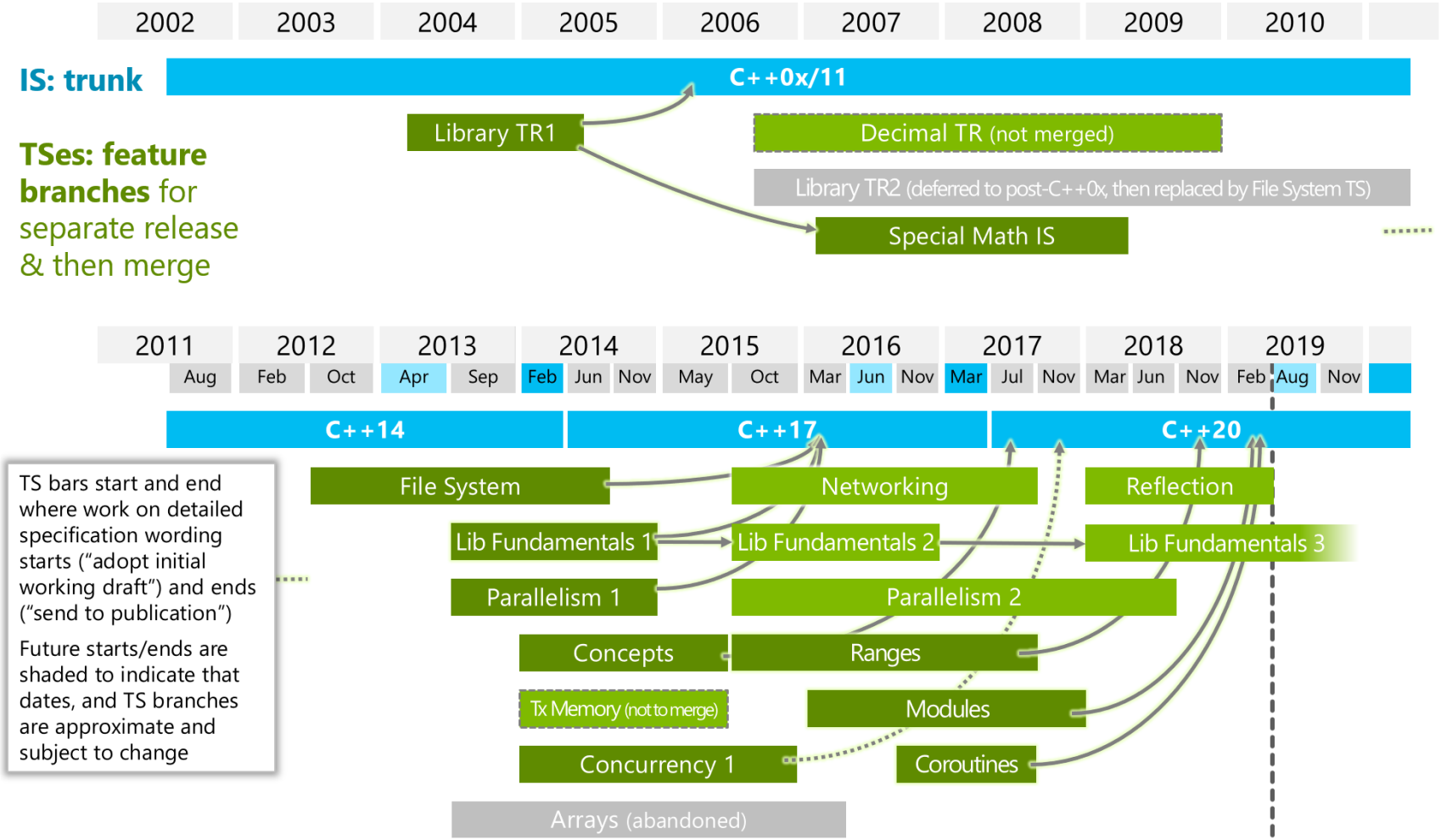
C++ - external libraries and OS interfaces

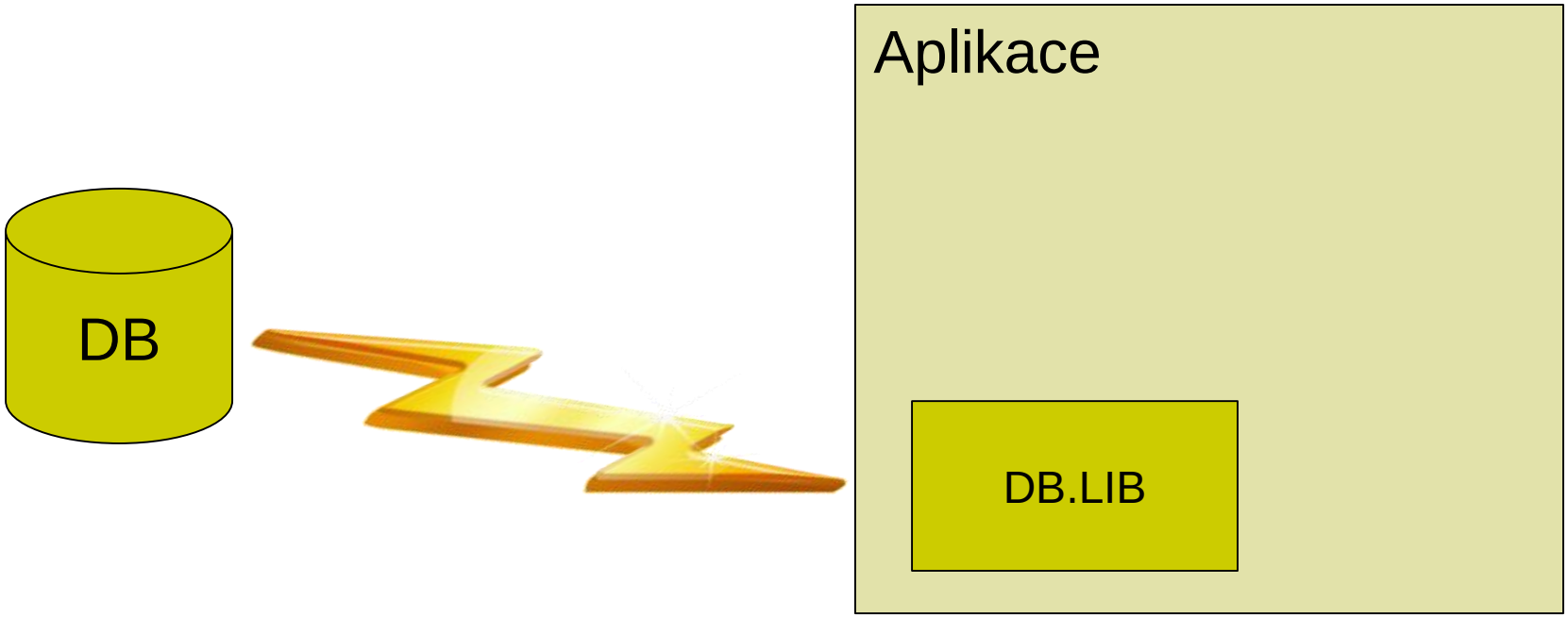
David Bednárek
Jakub Yaghob
Filip Zavoral

ISO C++ status


2002
2003
2004
2005
2006
2007
2008
2009
2010
IS: trunk
C++0x/11
Library TR1
Decimal TR (not merged)
TSes: feature branches for separate release & then merge
Library TR2 (deferred to post-C++0x, then replaced by File System TS)
Special Math IS
2011
2012
2013
2014
2015
2016
2017
2018
2019
Aug
Feb
Oct
Apr
Sep
Feb
Jun
Nov
May
Oct
Mar
Jun
Nov
Mar
Jul
Nov
Mar
Jun
Nov
Feb
Aug
Nov
C++14
C++17
C++20
File System
Networking
Reflection
Lib Fundamentals 1
Lib Fundamentals 2
Lib Fundamentals 3
Parallelism 1
Parallelism 2
Concepts
Ranges
Tx Memory (not to merge)
Modules
Concurrency 1
Coroutines
Arrays (abandoned)
TS bars start and end where work on detailed specification wording starts ("adopt initial working draft") and ends ("send to publication")
Future starts/ends are shaded to indicate that dates, and TS branches are approximate and subject to change

Database

- Client connection



Database – native client

- Oracle
  - OCI
- MS Server
  - ODBC, OLE DB
- Sybase
  - Open client
- PostgreSQL
  - Proprietary
- MySQL
  - Proprietary

Database – interface selection

- Native client
  - Faster
  - Specific features available
  - Not portable (sometimes)
  - Bound to specific DB
- General client (ODBC, OLE DB)
  - Implemented using native clients
  - Slower due to added layer
  - Specific features not accessible
  - Unbound to specific DB?

Database – API

- ISO 9075:2008
  - Almost all interfaces conform
    - Excluding MySQL
  - Defines a program entities hierarchy
  - Defines an interface for manipulating with program entities

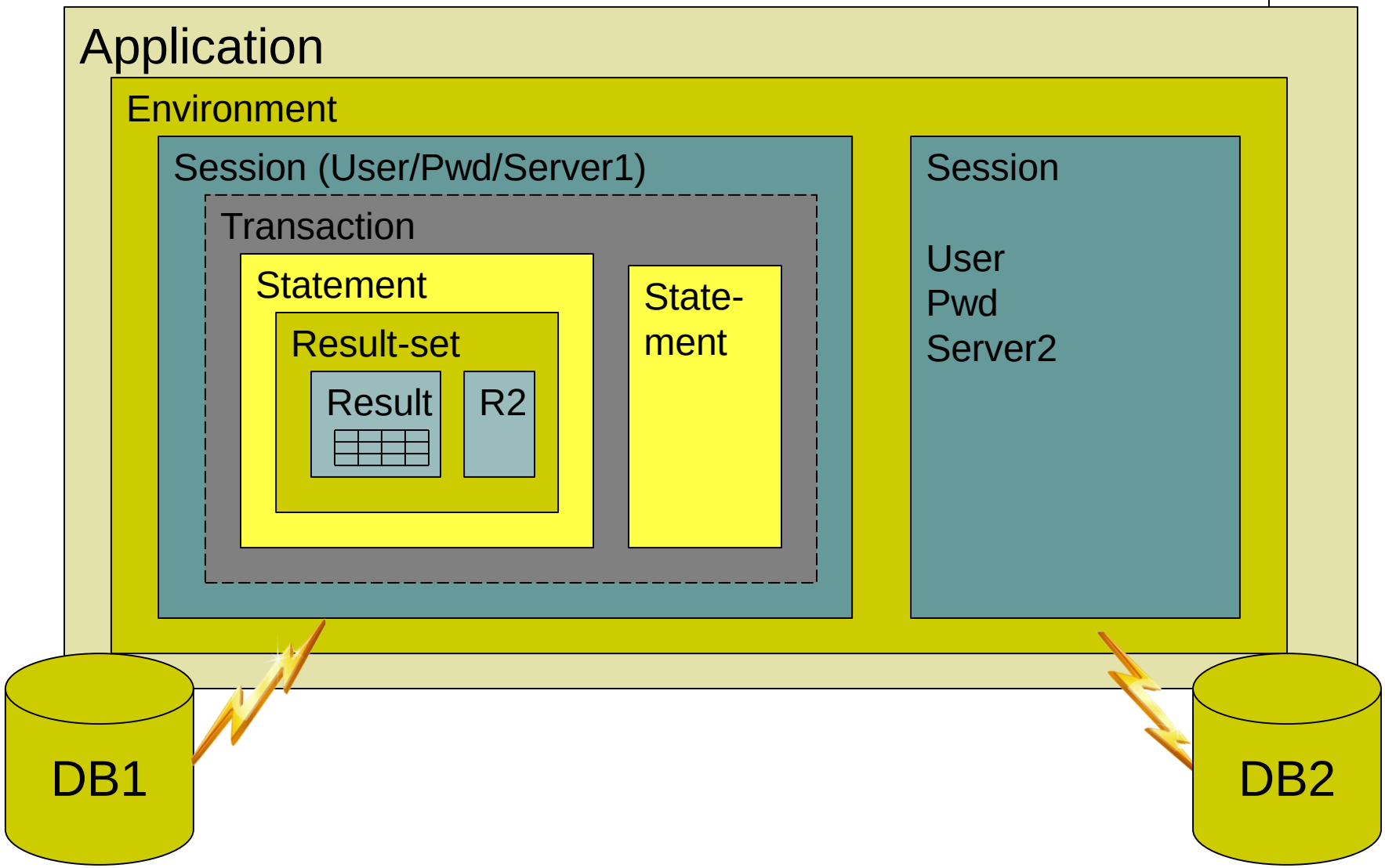
Database – program entities

- Environment
  - Global setting for client library in an application
  - Defines memory-management
- Session
  - Connection to one server
  - Properties session, user
  - Derived from the environment
- Statement
  - State space for one query
  - Prepared statement
  - Parameter binding
    - Static/dynamic, input/output
  - Derived from the session

Database – program entities

- Result-set
  - Container of results
  - Derived from the statement
- Result
  - Query result
  - Data binding
    - Static/dynamic
  - Derived from the result-set
- Transaction
  - Derived from the session

Database – program entities



Database – example

```
environment e;
CreateEnvironment(e);
session ses;
CreateSession(ses, e);
ConnectSession(ses, “user”,
    “pwd”, “srv”);
transaction t;
CreateTransaction(t, ses);
BeginTransaction(t);
statement stm;
CreateStatement(stm, ses);
int i, j;
BindParameter(stm, “q”, i, IN,
    C_INT, SQL_DEC);
result_set rs =
    ExecuteStatement(stm, “select
    * from t where c=:q”);
result r = rs.first();
BindColumn(r, 1, j, C_INT,
    SQL_DEC);
while(Fetch(r)) { }
DestroyResult(r);DestroyRS(rs);
CommitTransaction(t);
DestroyStatement(stm);
DestroyTransaction(t);
CloseSession(ses);
DestroySession(ses);
DestroyEnvironment(e);
```

Database – C++

- Something like SQLpp
  - Dead project

```
auto x = select(p.id, p.name, f.id.as(id2))
        .from(p, f)
        .where(p.name=any(select(f.name).from(f)))
        .group_by(f.name)
        .order_by(p.name.asc());
for(const auto & row : x)
{ int id = row.id; string name = row.name; }
```

Network

- C++23??
- BSD sockets
  - Winsock 2
- Socket
  - Abstract endpoint for network communication
  - Bind to different protocols
    - IPv4, IPv6
  - Bind to different kind of communication
    - Stream, datagram, raw
  - Bind to transportation protocol
    - TCP, UDP

Network – TCP client

```
SOCKET sock = socket(INET, STREAM, TCP);
struct sockaddr_in soain;
soain.sin_family = INET;
getaddrinfo(“www.nic.cz”, SRV_PORT, 0, &soain.sin_addr);
connect(sock, soain);
send(sock, buf, len);
recv(sock, buf, len);
shutdown(sock);
close(sock);
```

Network – TCP server

```
SOCKET listensock = socket(INET, STREAM, TCP);
struct sockaddr_in soain;
soain.sin_family = INET; soain.sin_port = SRV_PORT; soian.sin_addr
    = ANY;
bind(listensock, soain);
listen(listensock, queuedepth);
for(;;) {
    SOCKET sock = accept(listensock);              // blocks
    recv(sock, buf, len);   send(sock, buf, len);
    shutdown(sock);
    close(sock);
}
shutdown(listensock);
close(listensock);
```

Network – UDP client

```
SOCKET sock = socket(INET, DGRAM, UDP);
struct sockaddr_in soain;
soain.sin_family = INET;
getaddrinfo(“www.nic.cz”, SRV_PORT, 0, &soain.sin_addr);
sendto(sock, buf, len, &soain);
recvfrom(sock, buf, len, &soain);
close(sock);
```

Network – UDP server

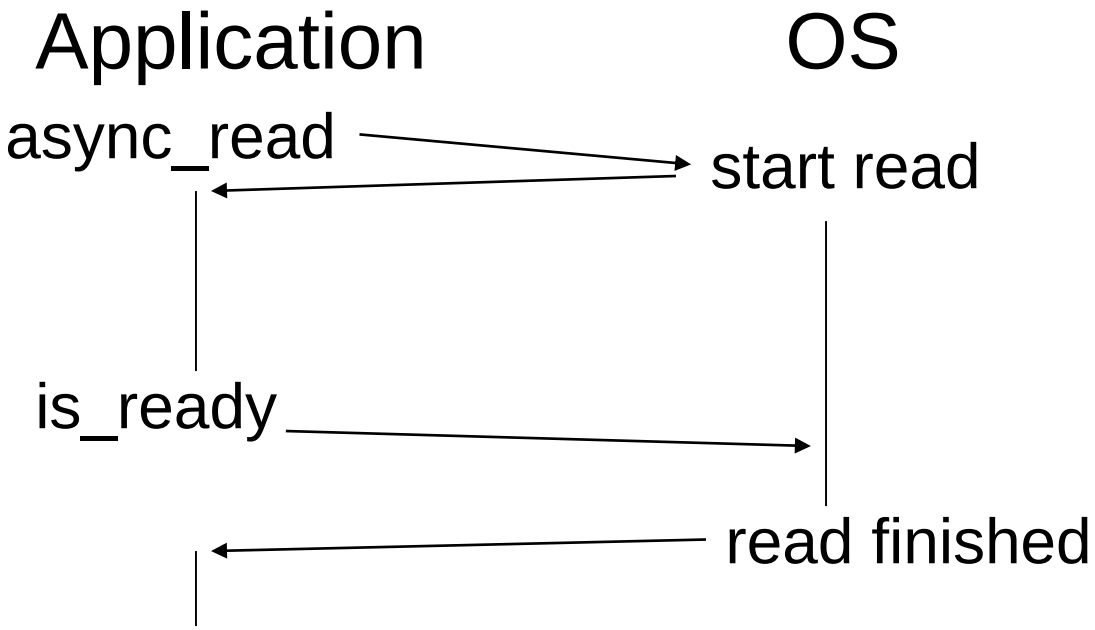
```
SOCKET sock = socket(INET, STREAM, TCP);
struct sockaddr_in soain;
soain.sin_family = INET; soain.sin_port = SRV_PORT; soian.sin_addr
    = ANY;
bind(sock, soain);
for(;;) {
    struct sockaddr_in soain_cli;
    int len = recvfrom(sock, buf, maxlen, &soain_cli);
    sendto(sock, buf, len, &soain_cli);
}
close(sock);
```

Network – remaining functions

- select
  - Wait for finished operations on sockets
- getaddrinfo
  - General translation from node name to a binary representation
- getnameinfo
  - General translation from binary representation to a node name and service name
- htons, htonl, ntohs, ntohl
  - Translation from host byte-order to network byte-order and vice versa

Asynchronous I/O

- Execute a file operation and continue in a work
  - A new entity created, which can be questioned for operation status or the application can wait for it
- What about coroutines?



Asynchronous I/O – demo

Windows

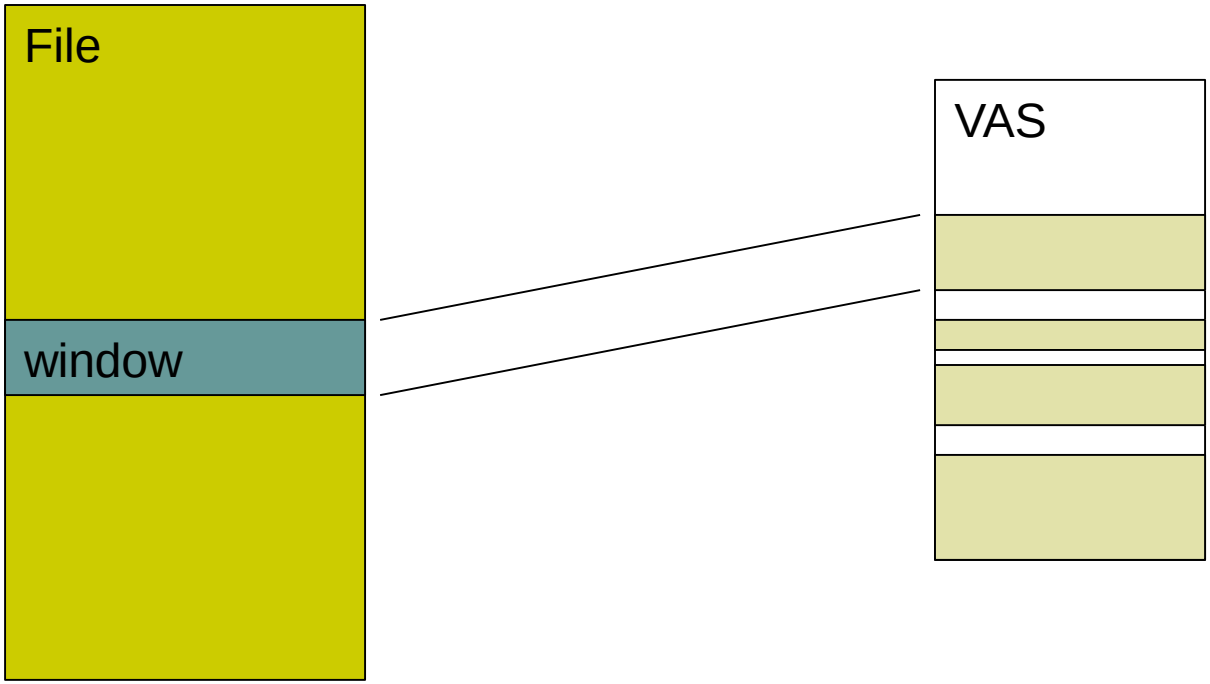
```
HANDLE f = CreateFile("f.txt",
    FILE_READ_DATA,
    FILE_SHARE_READ, 0,
    OPEN_EXISTING,
    FILE_FLAG_RANDOM_ACCESS |
    FILE_FLAG_OVERLAPPED, 0);
OVERLAPPED o;
memset(&o, 0, sizeof(OVERLAPPED));
o.hEvent = CreateEvent(0, TRUE, FALSE,
    0);
o.Offset = offs;
ReadFile(f, buf, sz, &rsz, &o);

WaitForSingleObject(o.hEvent, INFINITE);
CloseHandle(o.hEvent);
CloseHandle(f);
```

AIO

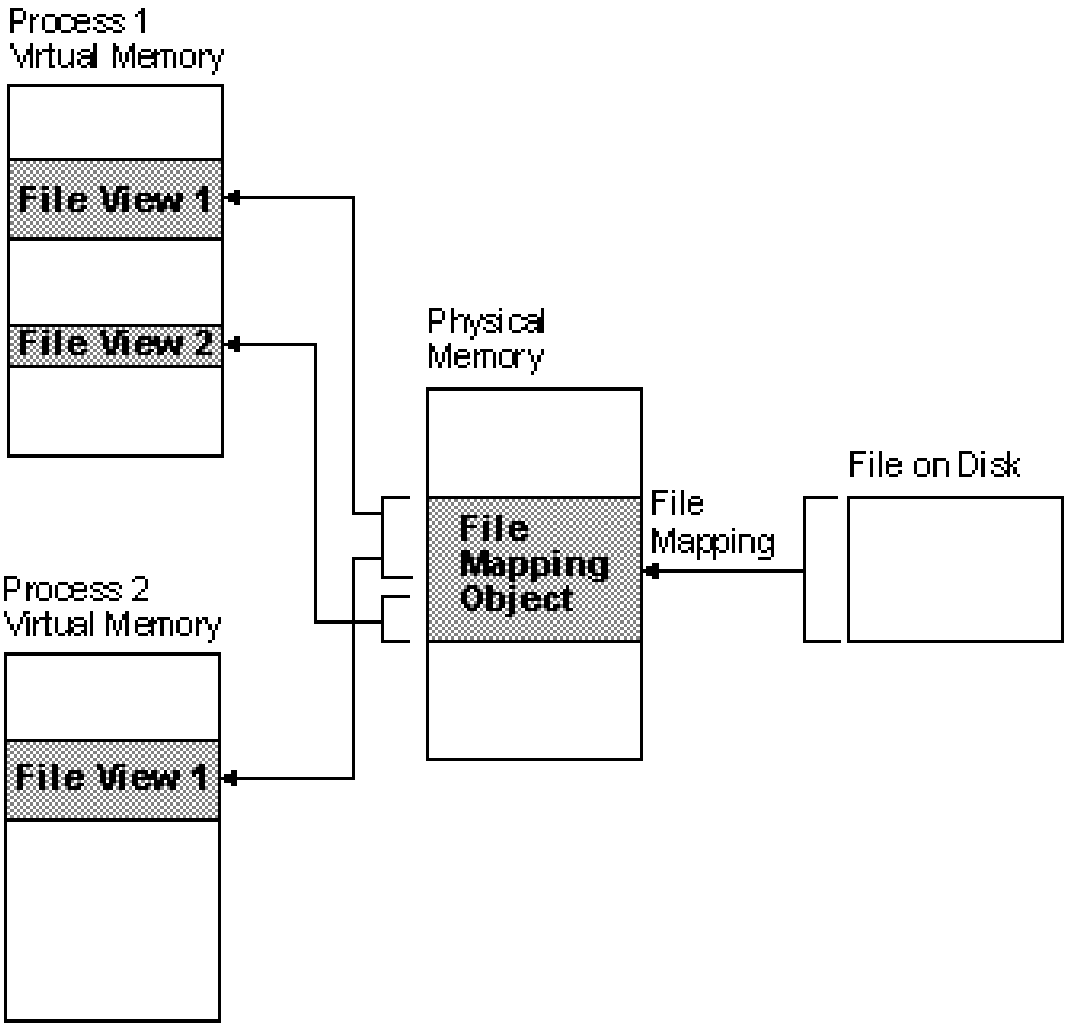
```
int f = open("f.txt", O_RDONLY |
    O_LARGEFILE, rights);
aiocb a;
memset(&a, 0, sizeof(aiocb));
aiocb. aio_fildes = f;
aiocb. aio_offset = offs;
aiocb. aio_nbytes = sz;
aiocb. aio_buf = buf;
aiocb. aio_sigevent.sigev_notify =
    SIGEV_NONE;
aio_read(&a);

aio_suspend(&a, 1, 0);
rsz = aio_return(&a);
close(f);
```

Memory mapped files

- Mapping, unmapping files
  - A file must be already opened
- Create, destroy window



Memory mapped files – Windows



Memory mapped files – demo

Windows

```
HANDLE f = CreateFile(“f.txt”,
    FILE_READ_DATA,
    FILE_SHARE_READ, 0,
    OPEN_EXISTING,
    FILE_FLAG_RANDOM_ACCESS, 0);
HANDLE m = CreateFileMapping(f, 0,
    PAGE_READONLY, 0, sz, 0);
void *p = MapViewOfFileEx(m,
    FILE_MAP_READ, 0, offs, sz, hinta);

UnmapViewOfFile(p);
CloseHandle(m);
CloseHandle(f);
```

POSIX

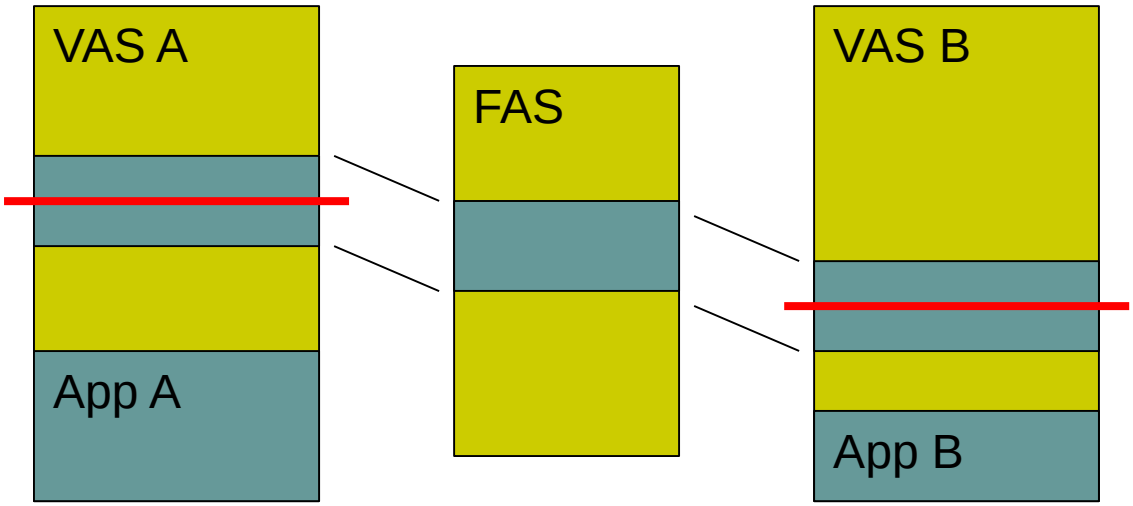
```
int f = open(“f.txt”, O_RDONLY |
    O_LARGEFILE, rights);
void *p = mmap(hinta, sz, PROT_READ,
    MAP_SHARED, f, offs);

munmap(p, sz);
close(f);
```

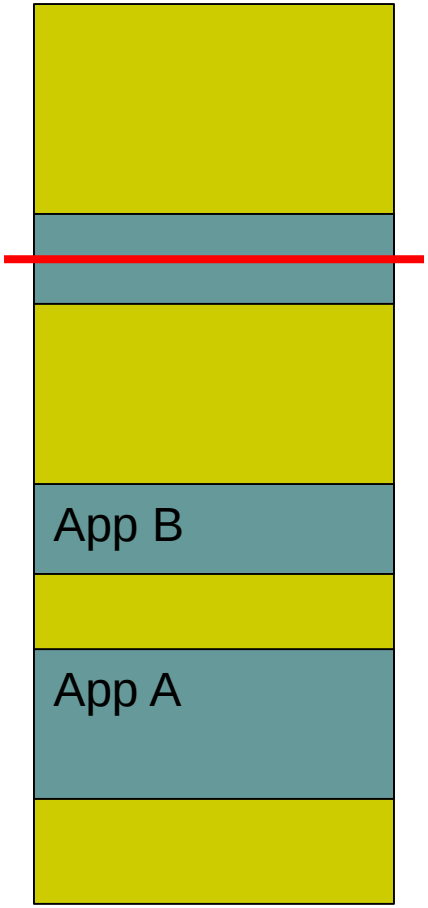
Filesystem

- Open, close directory
  - Some OS with name filtering
- Reading directory
  - Getting name, attributes, size, rights, …
- Create, remove directory
- Already in C++17

Shared memory

- Sharing memory among different processes
- Usually identified by a name
- Page granularity
- MASOS, SASOS
  - Offsets





Shared memory – demo SASOS

Windows client

```
HANDLE shm = OpenFileMapping(
    FILE_MAP_WRITE|FILE_MAP_READ,
    FALSE, “mymemory”);
void *p = MapViewOfFileEx(shm,
    FILE_MAP_READ, 0, 0, sz, hinta);
struct s { int n; int *ptr; };
for(int i=0;i<static_cast<s*>(p)->n;++i)
  cout << static_cast<s*>(p)->ptr[i];
UnmapViewOfFile(p);
CloseHandle(shm);
```

Windows server

```
HANDLE shm = CreateFileMapping
    (INVALID_HANDLE_VALUE, 0,
    PAGE_READWRITE, 0, sz,
    “mymemory”);
void *p = MapViewOfFileEx(shm,
    FILE_MAP_WRITE, 0, 0, sz, hinta);
struct s { int n; int *ptr; };
static_cast<s*>(p)->n = N;
static_cast<s*>(p)->ptr =
    static_cast<char*>(p)+sizeof(s);
for(int i=0;i<N;++i)
  static_cast<s*>(p)->ptr[i] = i;
UnmapViewOfFile(p);
CloseHandle(shm);
```

Shared memory – demo MASOS

Windows client

```
HANDLE shm = OpenFileMapping(
    FILE_MAP_WRITE|FILE_MAP_READ,
    FALSE, “mymemory”);
void *p = MapViewOfFileEx(shm,
    FILE_MAP_READ, 0, 0, sz, hinta);
struct s { int n; size_t offs; };
for(int i=0;i<static_cast<s*>(p)->n;++i)
  cout <<
    static_cast<int*>(static_cast<char*>(p)
    +static_cast<s*>(p)->offs)[i];
UnmapViewOfFile(p);
CloseHandle(shm);
```

Windows server

```
HANDLE shm = CreateFileMapping
    (INVALID_HANDLE_VALUE, 0,
    PAGE_READWRITE, 0, sz,
    “mymemory”);
void *p = MapViewOfFileEx(shm,
    FILE_MAP_WRITE, 0, 0, sz, hinta);
struct s { int n; size_t offs; };
static_cast<s*>(p)->n = N;
static_cast<s*>(p)->offs = sizeof(s);
for(int i=0;i<N;++i)
  static_cast<int*>(static_cast<char*>(p)
    +static_cast<s*>(p)->offs)[i] = i;
UnmapViewOfFile(p);
CloseHandle(shm);
```

Shared memory – demo

POSIX client

```
int shm = shm_open(“mymemory”,
    O_RDWR, rights);
void *p = mmap(hinta, sz, PROT_READ,
    MAP_SHARED, shm, 0);
struct s { int n; int *ptr; };
for(int i=0;i<static_cast<s*>(p)->n;++i)
  cout << static_cast<s*>(p)->ptr[i];
munmap(p, sz);
close(shm);
shm_unlink(“mymemory”);
```

POSIX server

```
int shm = shm_open(“mymemory”,
    O_RDWR|O_CREAT|O_EXCL,
    rights);
void *p = mmap(hinta, sz,
    PROT_READ|PROT_WRITE,
    MAP_SHARED, shm, 0);
struct s { int n; int *ptr; };
static_cast<s*>(p)->n = N;
static_cast<s*>(p)->ptr =
    static_cast<char*>(p)+sizeof(s);
for(int i=0;i<N;++i)
  static_cast<s*>(p)->ptr[i] = i;
munmap(p, sz);
close(shm);
shm_unlink(“mymemory”);
```

POSIX

- Portable Operating System Interface [for Unix]
  - Existing native implementation for Win NT
  - Single UNIX Specification
  - IEEE standard
  - API
  - Shell
  - Utility

POSIX – versions

- POSIX.1
  - API (includes ANSI C)
  - 1988
- POSIX.1b
  - Real-time extension
  - 1993
- POSIX.1c
  - Thread extension
  - 1995
- POSIX.2
  - Shell and utility
  - 1992
- POSIX:2001
  - API, shell, utility
- POSIX:2004
  - Update and corrigendum
- POSIX:2008
  - API, shell, utility

POSIX – compatibility

- Full
  - AIX, HP-UX, Solaris, Mac OS X, …
- Partial
  - FreeBSD, GNU/Linux, OpenSolaris, …
- Windows
  - Cygwin – partial compatibility
  - MS Windows Services for UNIX 3.5 – full compatibility

Boost

- Portable C++ library
  - Review process
  - Source and testing ground for ISO
    - Tightly coupled with ISO comittee
  - Large set of supported OSs and compilers
  - Very good library, sometimes too abstract

Boost – containers

- Array
  - Constant size array
  - ISO C++ TR1/C++ 11
- Bimap
  - Two-sided map
- Circular buffer
- Graph
- Variant
  - Secure union
  - ISO C++17

Boost – parallel programming support

- Interprocess
  - Shared memory
  - Memory mapped files
  - Mutexes
    - ISO C++ 11
- MPI
- Thread
  - ISO C++ 11

Boost – I/O

- Asio
  - Asynchronous I/O, executors, networking, …
  - Parts maybe in C++20?
- Format
  - Enhanced formatting like printf
- Program options
  - Command line parameters
- Serialization
  - Data serialization for saving or marshalling

Boost – mathematic and numeric

- Intervals
- Multi-array
- Random
  - ISO C++ TR1/C++ 11

Boost – generic programming and metaprogramming

- GIL
  - Generic Image Library
- Static assert
  - ISO C++ 11
- Type traits
  - ISO C++ TR1/C++ 11
- MPL
  - General framework for compile-time algorithms, sequences, and metafunctions

Boost – strings

- Lexical cast
- Regex
  - ISO C++ TR1/C++ 11
- Spirit
  - LL parser
- String algo
- Tokenizer
  - Splitting a string to tokens

Boost – unsorted

- Shared ptr
  - ISO C++ TR1/C++ 11
- Date time
  - ISO C++ 11
- Filesystem
  - Paths, directory, files
  - ISO C++17