# Stream Objects

Peter J. Denning

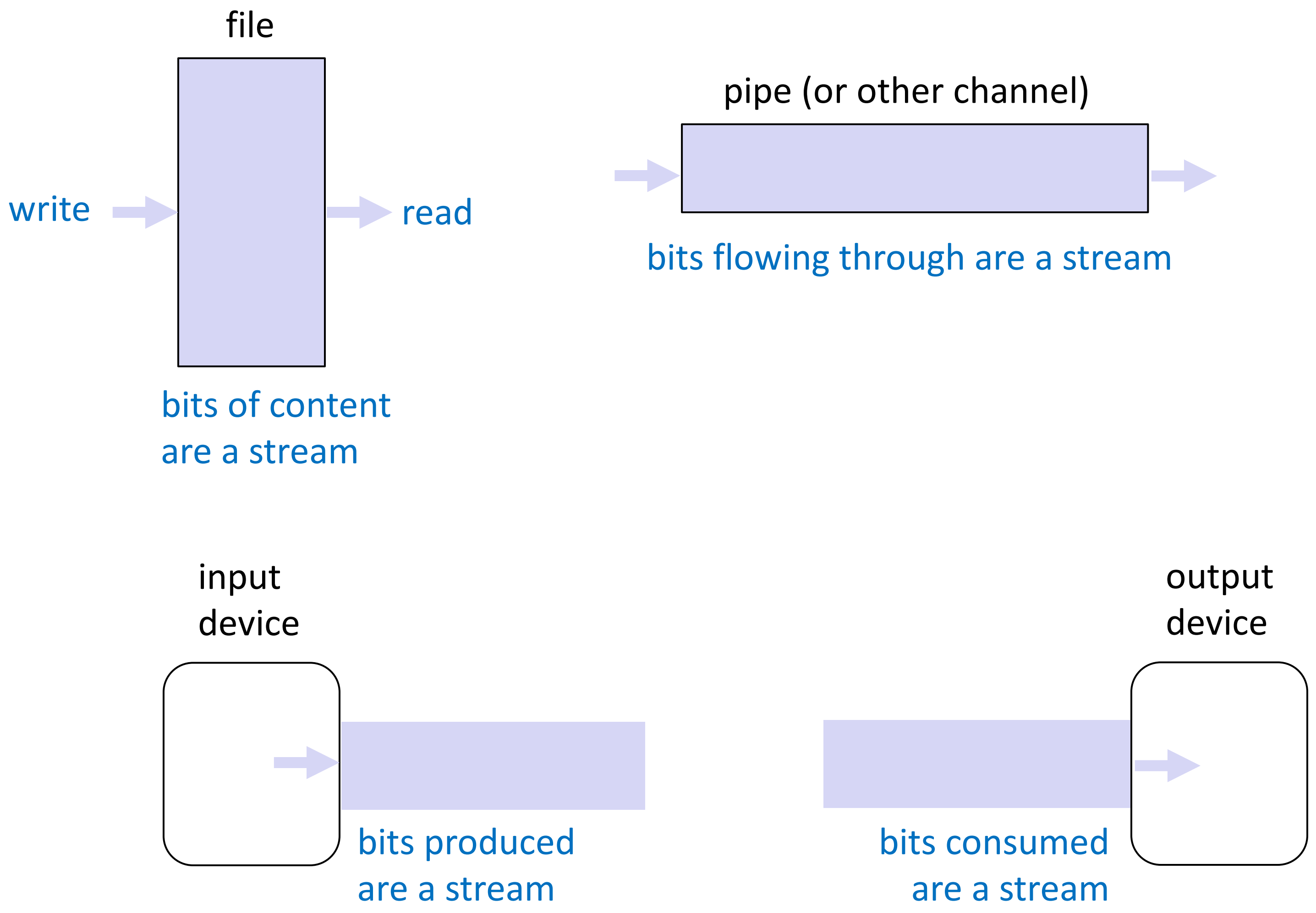
## Two kinds of information object

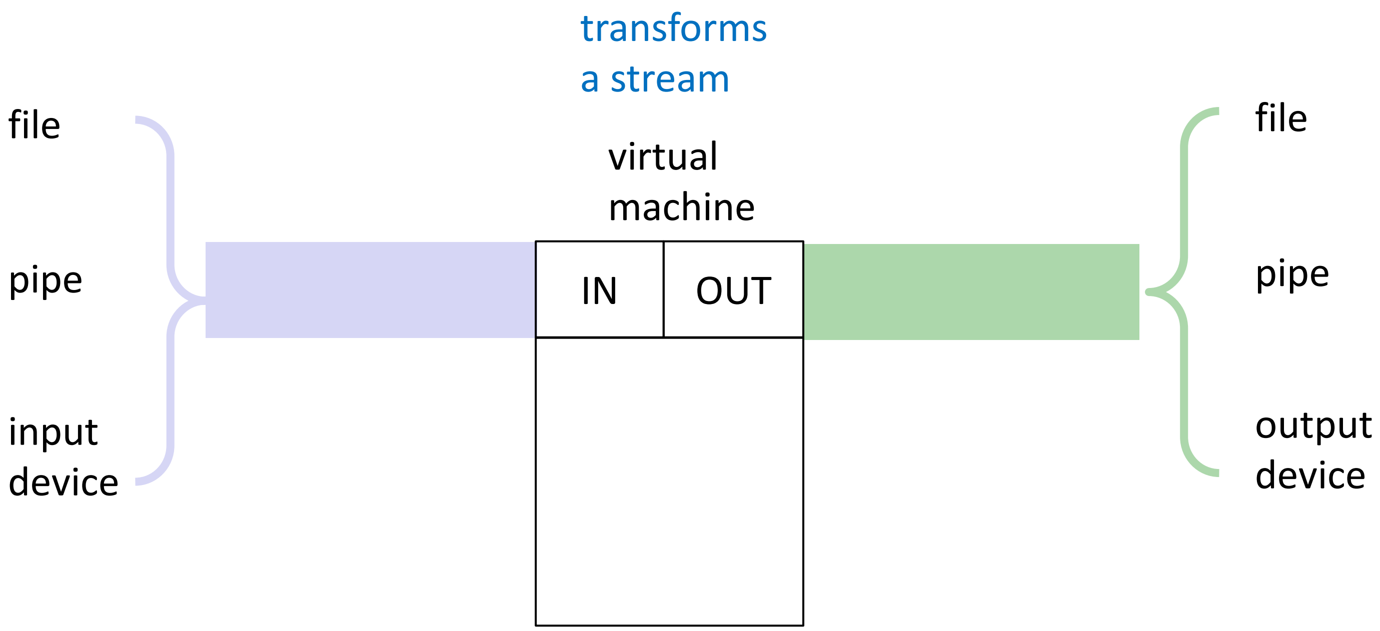
- Streams
- Directories

- (Directories considered in next module.)

## Streams

- Contents of file
- Flow on a channel
- Flow produced by input device
- Flow consumed by output device


file
write
read
bits of content are a stream
pipe (or other channel)
bits flowing through are a stream
input device
bits produced are a stream
output device
bits consumed are a stream

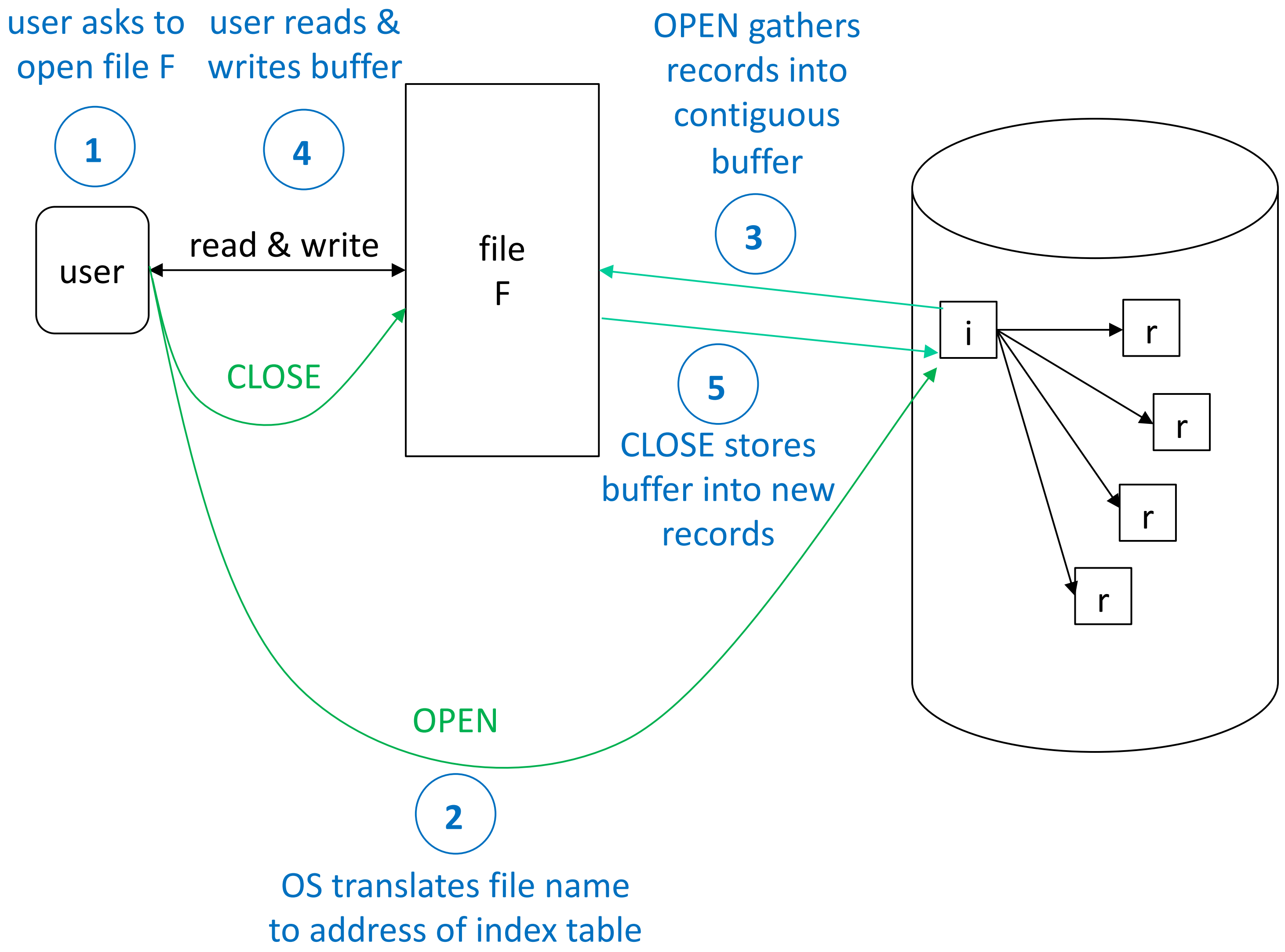


Streams allow great flexibility in constructing pipelines of virtual machines

## Files

- Named object containing a stream of bits of definite length
- Complex structure when stored on disk
  - File divided into equal size “records”
  - Index table points to all the records
  - File disk address = address of index table
- Need OPEN command to create file image in main memory buffer for reading and writing.

- Problem: performance of direct read or write would be unacceptable
  - One disk access for each record
  - Modified file may require overwriting of existing records
- Solution: copy the records into a contiguous file image in a main memory buffer
  - Reads and writes access the buffer
  - Disk accesses only when file opened or closed


user asks to open file F
1
user reads & writes buffer
4
OPEN gathers records into contiguous buffer
3
user
read & write
file F
CLOSE
i
r
r
r
r
5
CLOSE stores buffer into new records
OPEN
2
OS translates file name to address of index table

## File system interface

- fc = CREATE_FILE( )
  - create index record with empty tree; create file capability fc pointing to it
- DELETE_FILE(fc)
  - remove index record fc and other records
- These commands can be done by any process

## File system interface - 2

- bc = OPEN_FILE(fc)
  - create buffer capability bc pointing to buffer; tag buffer with fc; copy records in order into buffer; return bc to user
- CLOSE_FILE(bc)
  - copy contents of buffer as set of records to disk with index table fc, replacing old records; delete buffer
- Most often used by shell when providing IN and OUT pointers to virtual machines

# File system interface - 3

- (A,L) = READ_FILE(bc)
  - Copy buffer contents as stream of length L to address space beginning at address A
- WRITE_FILE(bc,(A,L))
  - copy stream (A,L) from address space to buffer, reset buffer size to L
- Parent of virtual machines puts the bc capabilities in the IN or OUT ports

## Pipes

- Flow of bits from sender to receiver
- Indefinite length
  - sender must append EOS (end of stream) special symbol to mark the end
- Channel synchronized so that sender must wait when channel full and receiver when it is empty (the producer-consumer relationship)

## Pipes Interface

- pc = CREATE_PIPE( )
  - create a new pipe named by pipe capability pc
  - set up the finite buffer to hold its contents
- DELETE_PIPE(pc)
  - delete the pipe and buffer named by pc

## Pipes Interface - 2

- pcr = OPEN_PIPE(pc,read)
  pcw = OPEN_PIPE(pc,write)
  - open the producer end for writing and return pipe write capability pcw (has its w bit on), or open the consumer end for reading and return pipe read capability pcr (has its r bit on)
- CLOSE_PIPE (pc)
  - close the pipe and its buffer

## Pipes Interface - 3

- L = READ_PIPE(pcr,A,m)
  - read L ≤ m bytes from the pipe (P-C synchronization), leaving stream (A,L) in caller’s address space; L may be <m if EOS encountered
- WRITE_PIPE(pcw,(A,L))
  - copy stream (A,L) from caller’s address space to the pipe (P-C synchronization)

## Devices

- Separate pieces of hardware that generate streams (input) or consume streams (output)
- Devices generally deal with fixed-size chunks of streams, leaving assembly into full stream to a higher-level process
- Drivers are a major source of bugs.  Therefore, OS associates devices as private objects of certain service processes, where proper use of the interface can be assured.

## Device interface

- dc = CREATE_DEVICE(device-driver-file)
  - Install the given file and return a device capability pointing to it
- DELETE_DEVICE(dc)
  - Uninstall the driver named by dc

## Device interface - 2

- dbc = OPEN_DEVICE(dc)
  - Create a buffer to hold device stream chunks, return capability pointing to it
- CLOSE_DEVICE(dbc)
  - Delete the open buffer dbc
- READ_DEVICE(dbc) and WRITE_DEVICE(dbc)
  - move one chunk between service process and device; service process gathers chunks into streams

## Common Interface

- Generic interface for CREATE, DELETE, OPEN, CLOSE, READ, WRITE permits virtual machines to read IN and write OUT regardless of the type of stream object IN and OUT are connected to

## Common Interface - 2

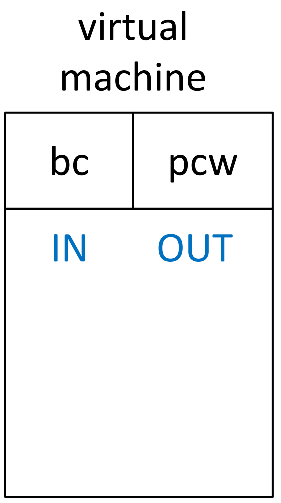
- Generic CREATE and DELETE
- c=CREATE(X)
  - X is the type of stream object (file, pipe, device)
  - create an empty version of the object and capability c pointing to it
- DELETE(c)
  - delete the object pointed to by c

## Common Interface - 3

- Generic OPEN and CLOSE
- b=OPEN(c)
  - X = c.type (file, pipe, device)
  - route call to OPEN_X(c)
  - return b, capability pointing to the buffer
- CLOSE(b)
  - X = type of object in buffer b
  - route call to CLOSE_X(b)

## Common Interface - 4

- Generic READ and WRITE
- (A,L) = READ(bc)
  - X = type of object in buffer bc
  - route call to READ_X(bc)
- WRITE(bc,(A,L))
  - X = type of object in buffer bc
  - route call to WRITE_X(bc,(A,L))



In the IN port, the parent of this VM provided a buffer capability bc for an opened file. The READ(IN) command within the virtual machine calls READ_FILE(bc) on seeing that the type of the IN capability is for an open file buffer.

In the OUT port, the parent of this VM provided an open pipe write capability pcw. The WRITE(OUT) command calls WRITE_PIPE(pcw,(A,L)) on seeing that the type of the OUT capability is for the producer end of a pipe.

## Summary

- Streams: the abstraction used by files, pipes, and devices
- Allows for a generic interface within VMs that reads (writes) from (to) any stream
- The parent of the VM simply inserts appropriate stream capabilities into the VM's IN and OUT ports