# Virtual Machine Basics

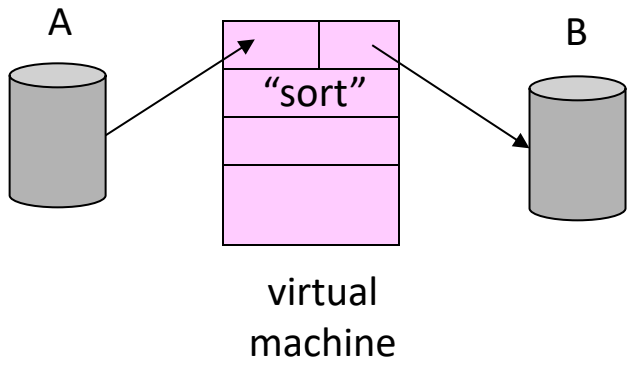
P. J. Denning

- User types command at command line interface
- OS starts process running the command code
- Provides input and output pointers

EXAMPLE:

sort A into B

create virtual machine
running “sort” code,
with file A as input and B
as output

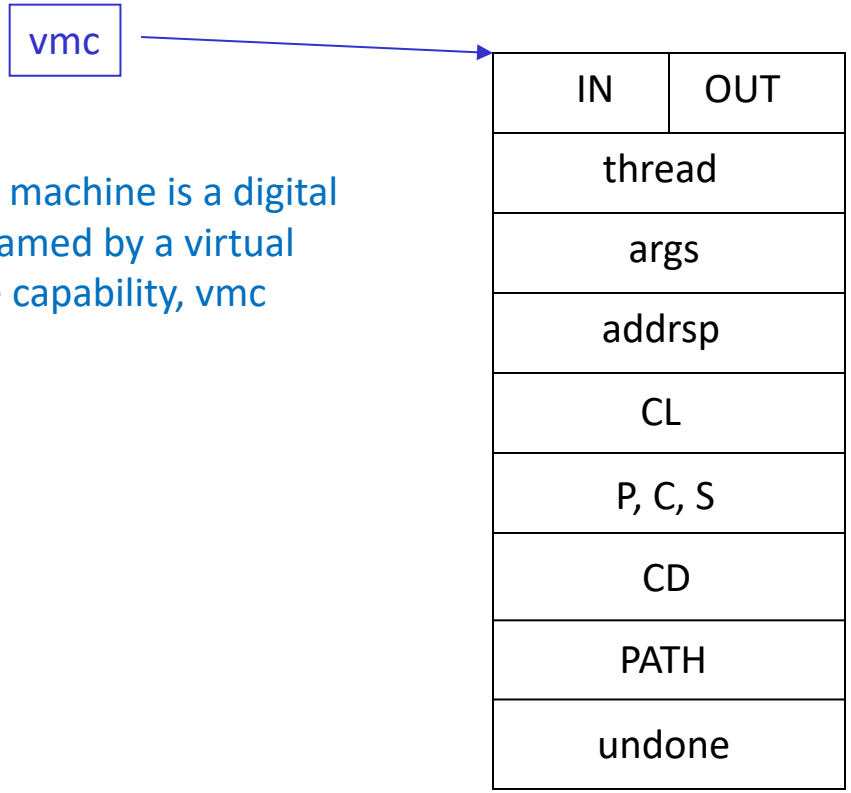


- Virtual machine is the standard structure that runs a process
  - "process is a program running on a virtual machine"
- Every command maps into one or more virtual machines

- Be aware that “virtual machine” has multiple meanings in the OS world:
  - Exact replica of hardware within a partition of memory (IBM VMS)
  - Simulation of one machine on another (Virtual PC)
  - Abstract machines (level structured OS)
  - Standard form for program-in-execution (Unix)
- For the kernel, we focus on the fourth meaning.

# What is a virtual machine?

- A simulated computational environment for a program, consisting of
  - An IN and OUT port
  - A thread running a program
  - A list of arguments (args)
  - An address space holding code, data, stack
  - A capability list
  - Parent, children, and sibling pointers
  - A current directory pointer
  - A current set of search pathnames
  - A count of undone children

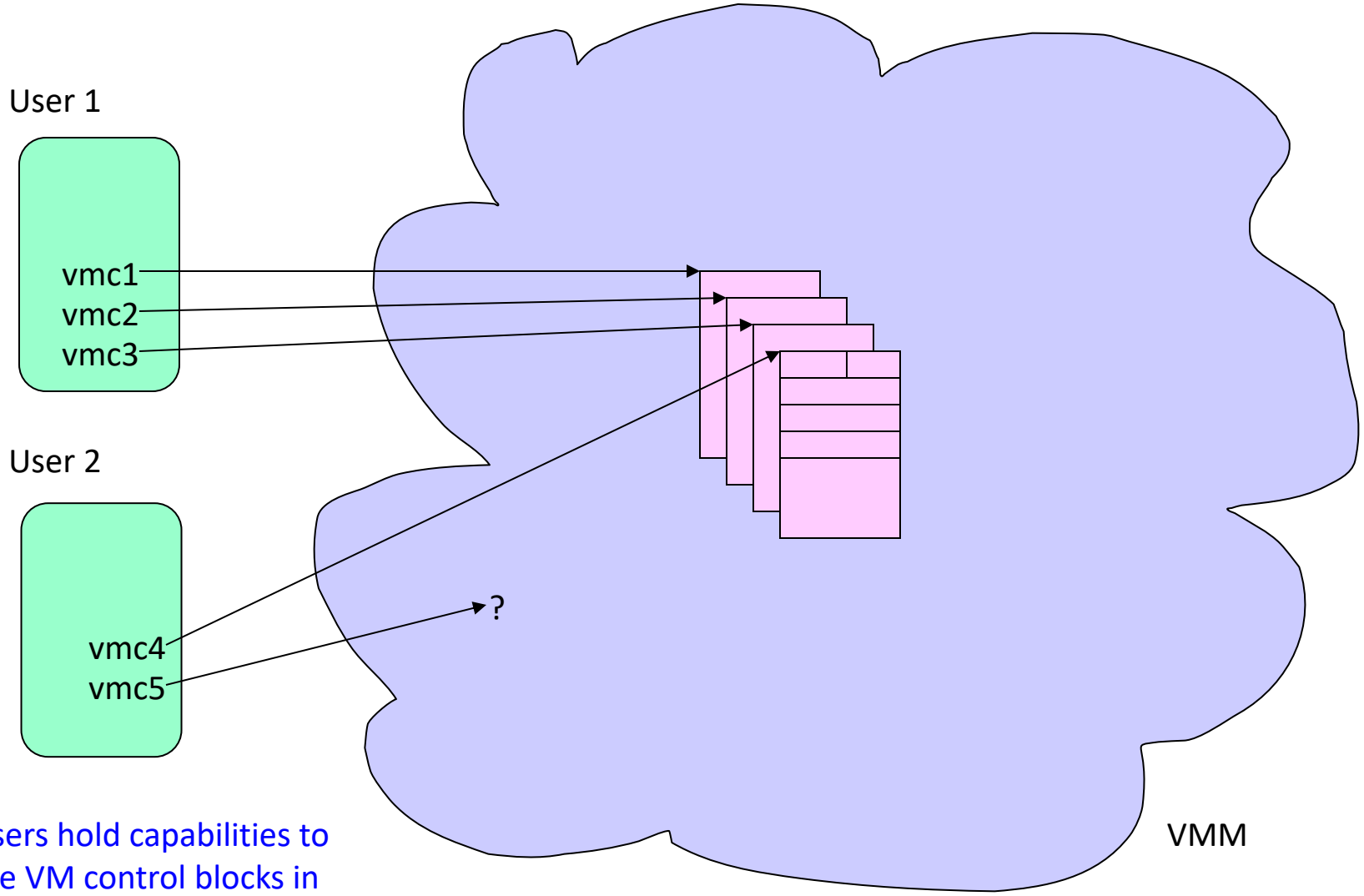



A virtual machine is a digital object named by a virtual machine capability, vmc

The VM itself is represented by a VM control block formatted according to this template

# Virtual Machine Manager (VMM)

- A subsystem that manages all VMs
- Gives each VM a ***capability*** on creation
  - (V, access, handle)
- Users present VM capabilities with requests to perform operations on the VMs named by the handles in those capabilities




Users hold capabilities to the VM control blocks in their C-lists (part of VM)

vmc5 is a dead capability left over from prior delete

# Interface

- vmc = CREATE_VM(initial values)
  - initial values list specifies value for each slot in template
  - thread in the vm is initially suspended
- DELETE(vmc)
  - delete VM and all its components
- COMPUTE(vmc)
  - move the VM thread to the ready list for execution
- EXIT

  final instruction of VM thread, “VM has finished its job”

# Execution Model

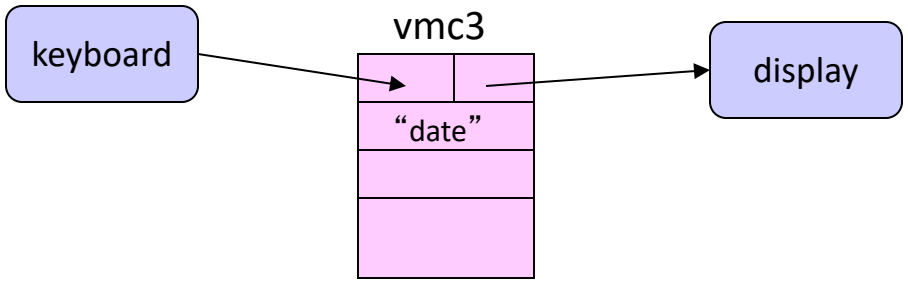
- Parent creates one or more children VMs, all initially in a suspended state
- Parent interconnects VMs with pipes, attaches source object to pipeline's IN port and sink object to OUT port
- Parent issues COMPUTE(vmc) command, where vmc is the capability for the first child linked to a list of siblings. This command starts all children, sets UNDONE = children count, and puts parent to sleep
- Each child completes with EXIT, which deducts 1 from UNDONE count of parent and awakens parent when UNDONE = 0

- This model ensures that parent and children (as group) are mutually excluded in their executions
- Therefore, no race conditions between children and parent for use of parent's standard IN and OUT

# Command Interpreter (Shell)

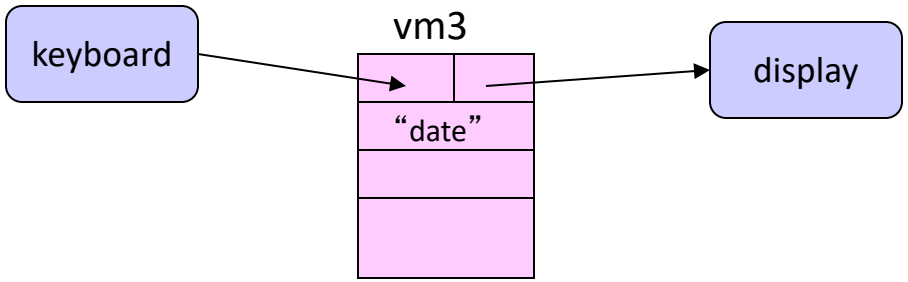
- Listen for user to type line of text
- Using parser, decompose text into substrings, each corresponding to four-part command component
- Create the VMs to carry out the pipe-connected commands
- With COMPUTE command set the VMs in motion; wait for all its children to EXIT
- Clean up (DELETE the VMs)
- Generate command prompt and repeat

# date




Each arrow represents a capability. Thus the IN port of vmc3 contains a capability (dev, r, handle) and OUT port contains (dev, w, handle)

# date

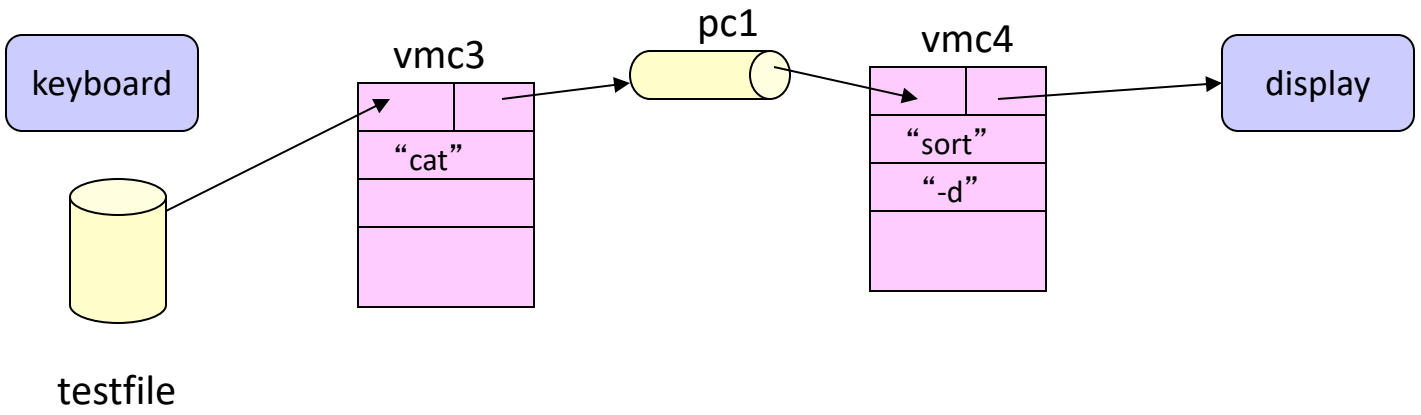



By default, a new VM inherits the standard IN and OUT of its parent.

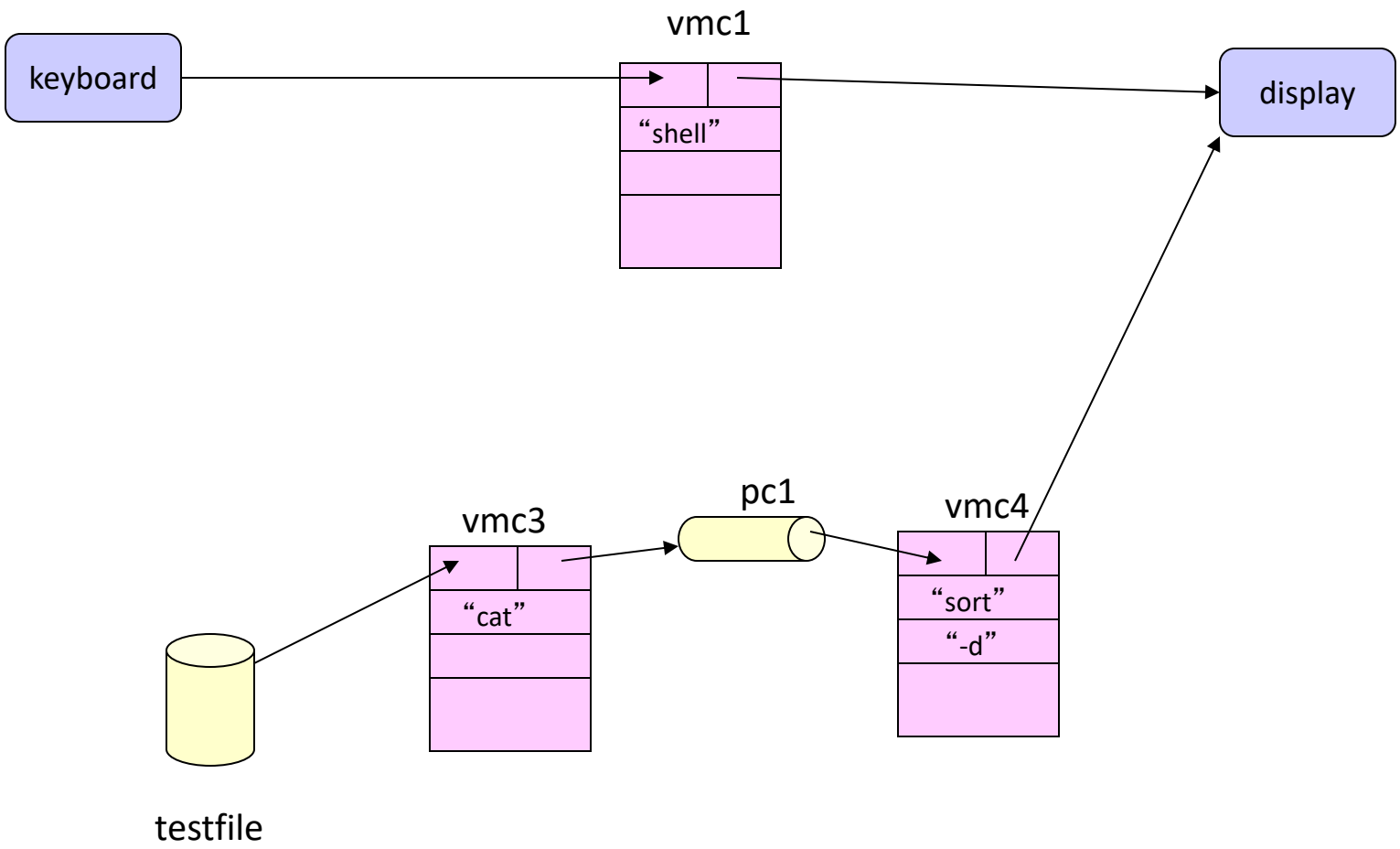
Typically IN = keyboard and OUT = display.

If different IN or OUT is needed, shell tells the VMM what sources or sinks to use.

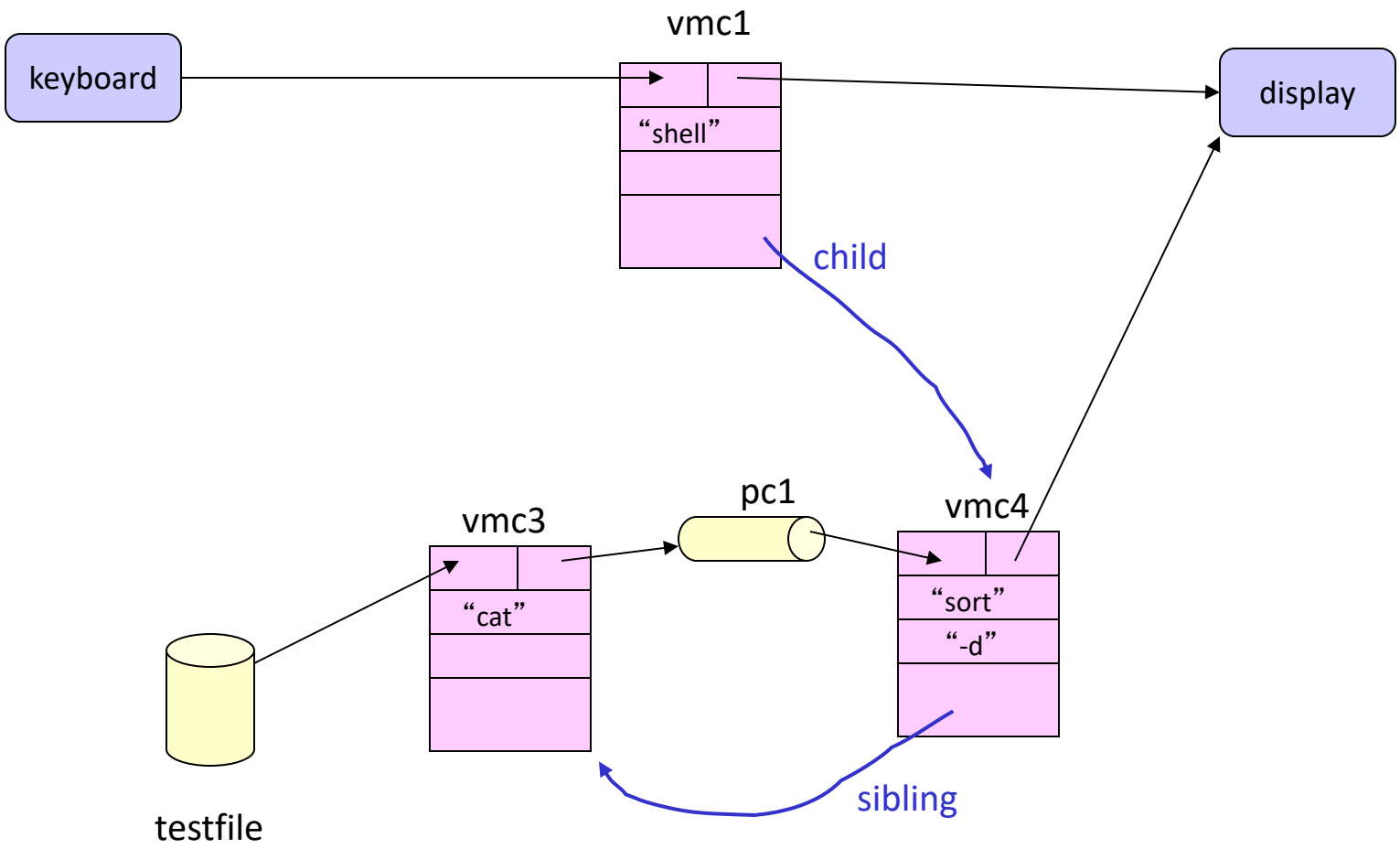
```
cat < testfile | sort -d
```



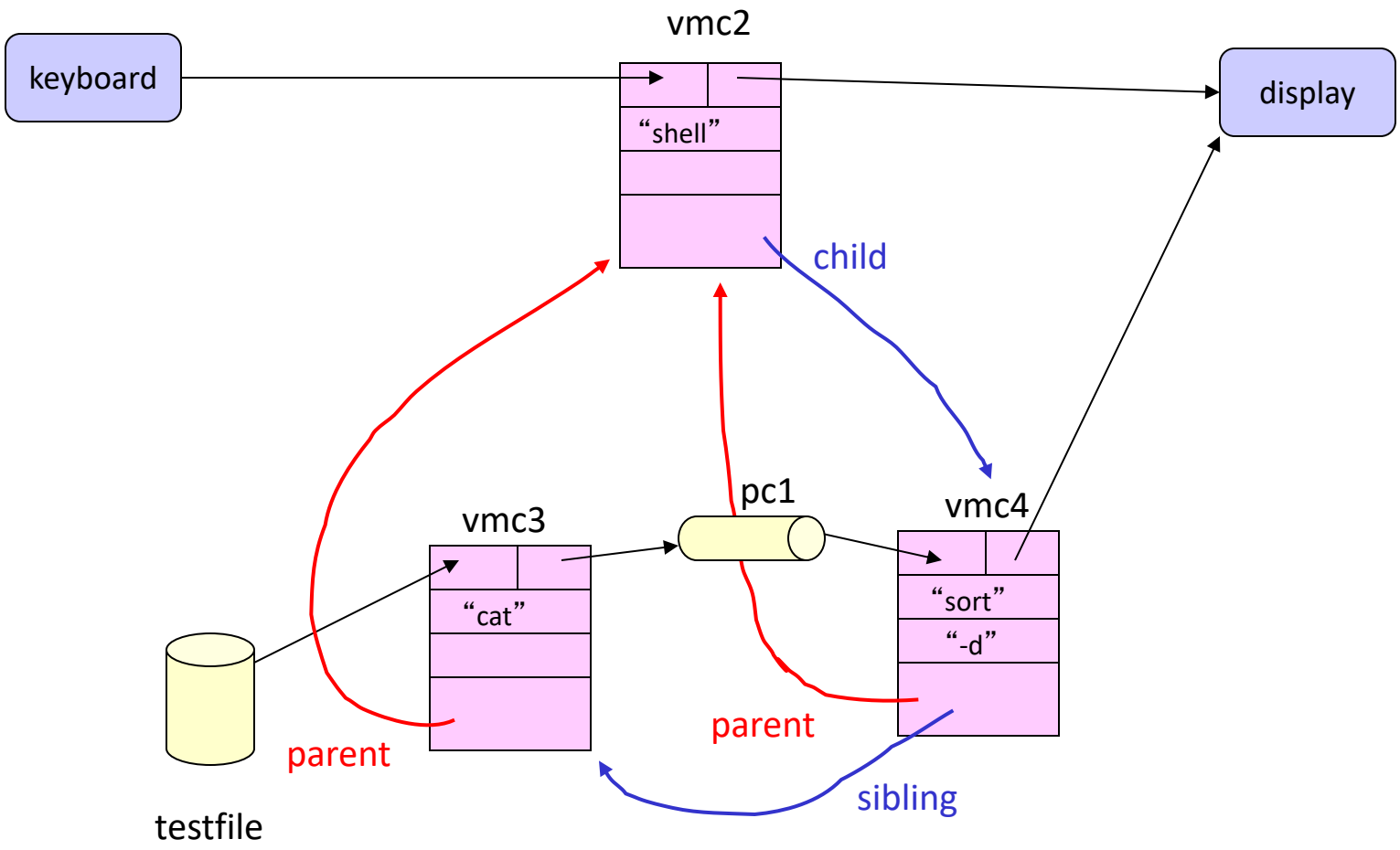
# `cat < testfile | sort -d`

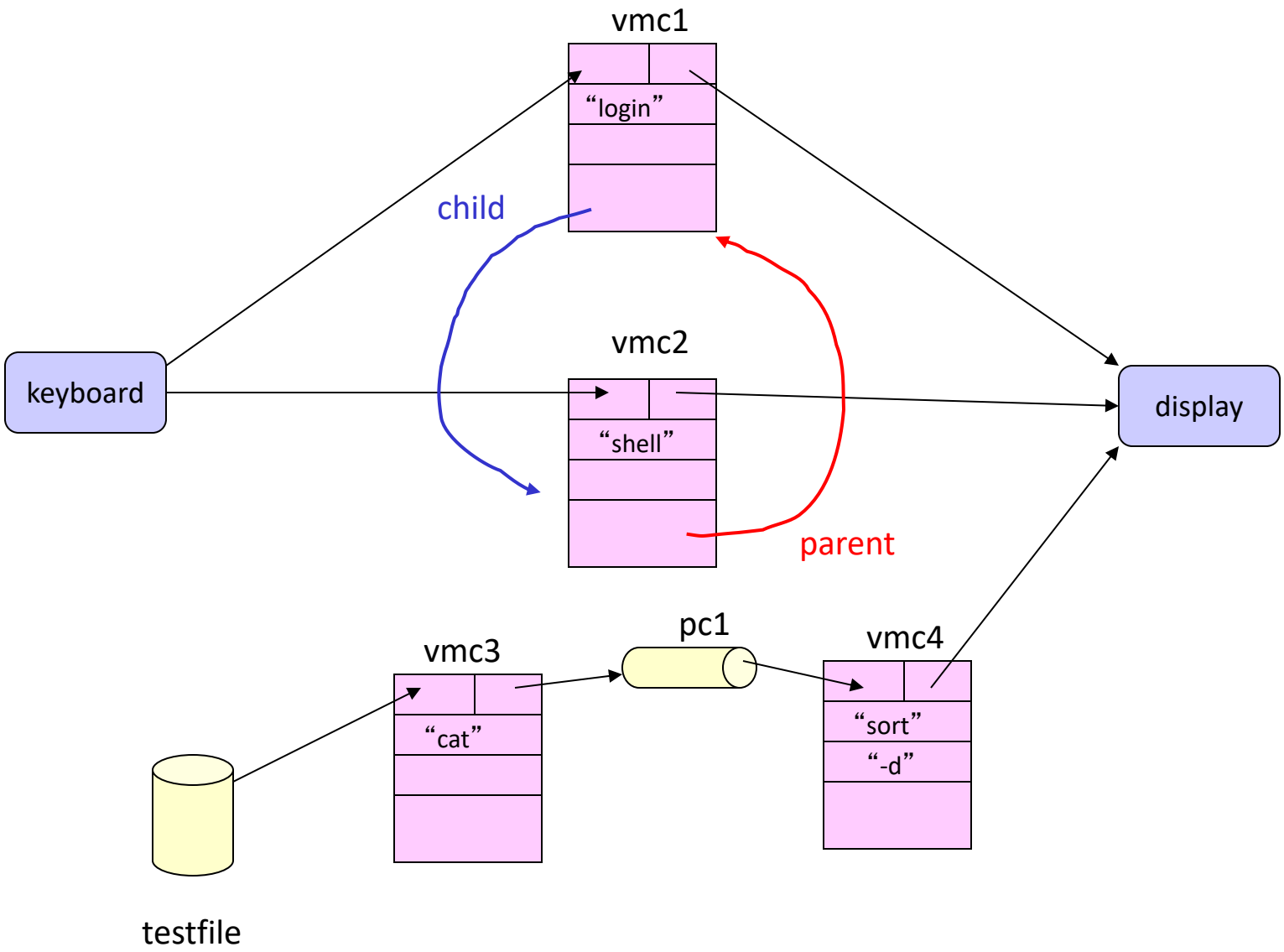


# `cat < testfile | sort -d`



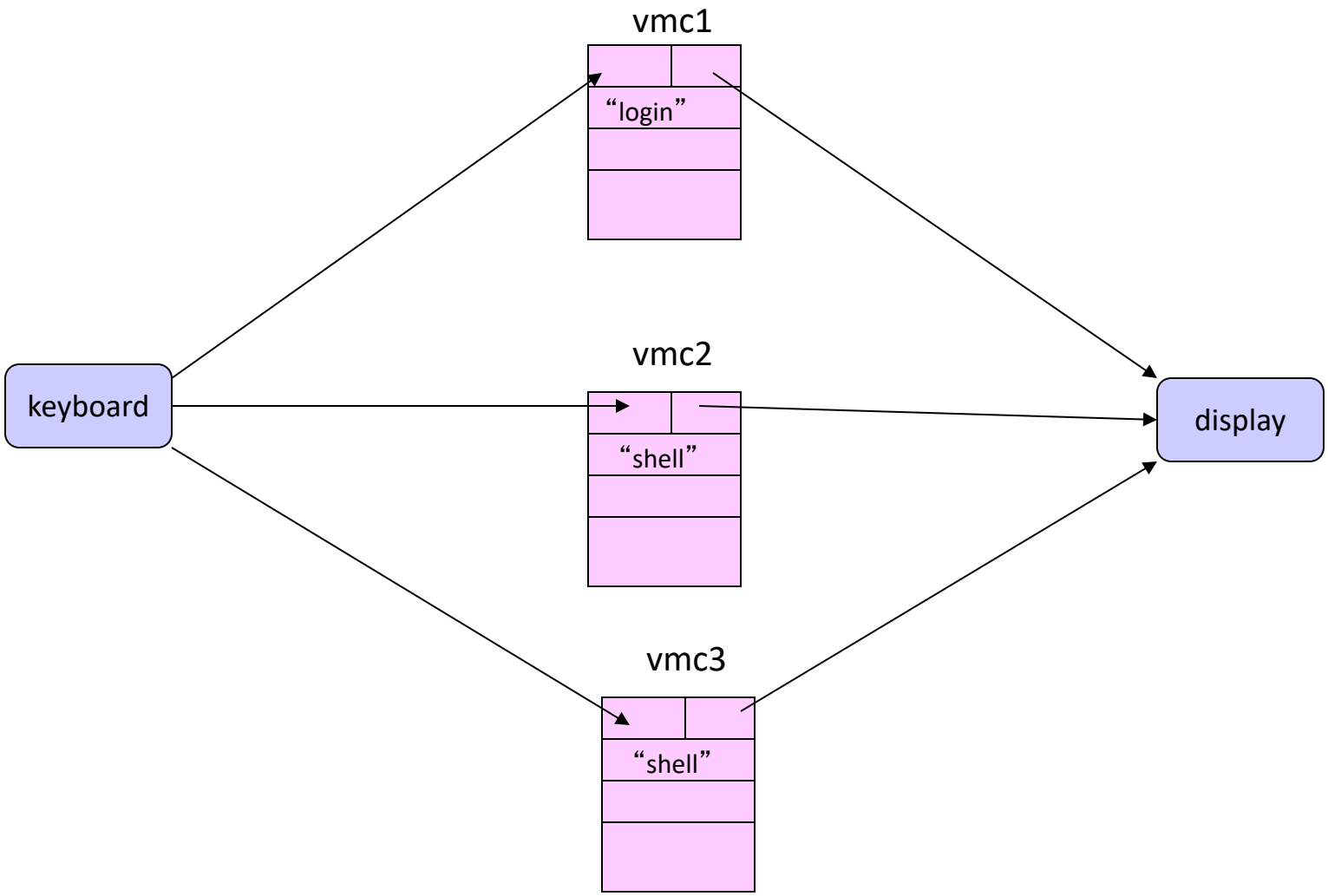
# `cat < testfile | sort -d`





# Recursive Shell

- “shell” is an allowable command --- creates a new shell VM as child to the issuer
- “login” is an allowable command; logs out old session (entire subtree from the login) and starts over



# Summary

- Virtual machine is a simulation of a standard execution environment of any process
- Parent virtual machine creates a group of children to execute a command and waits for all to finish before resuming
- All the components of a virtual machine are implemented at lower levels of the OS