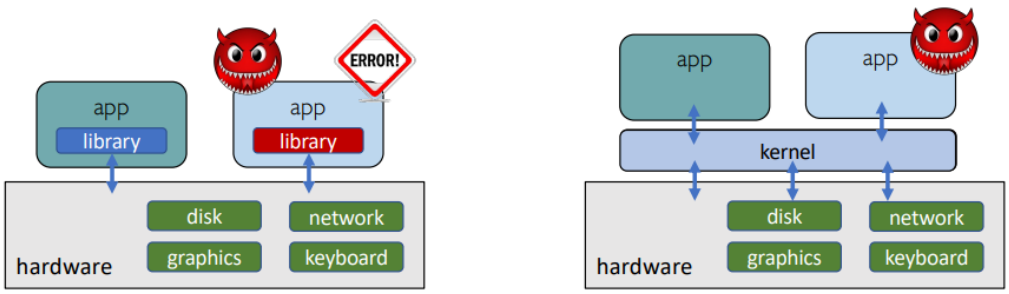
Lecture2 OS Basics

1. Dual-mode Operations

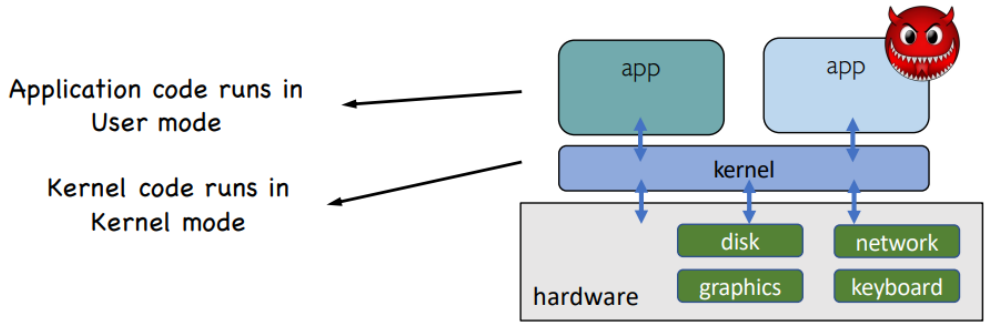
Evolution of Operation Systems



Kernel: A **bigger "library"** to handle low-level I/O

- Issue: Fault and security isolation
- **Kernel needs to be protected** from faulty/malicious apps

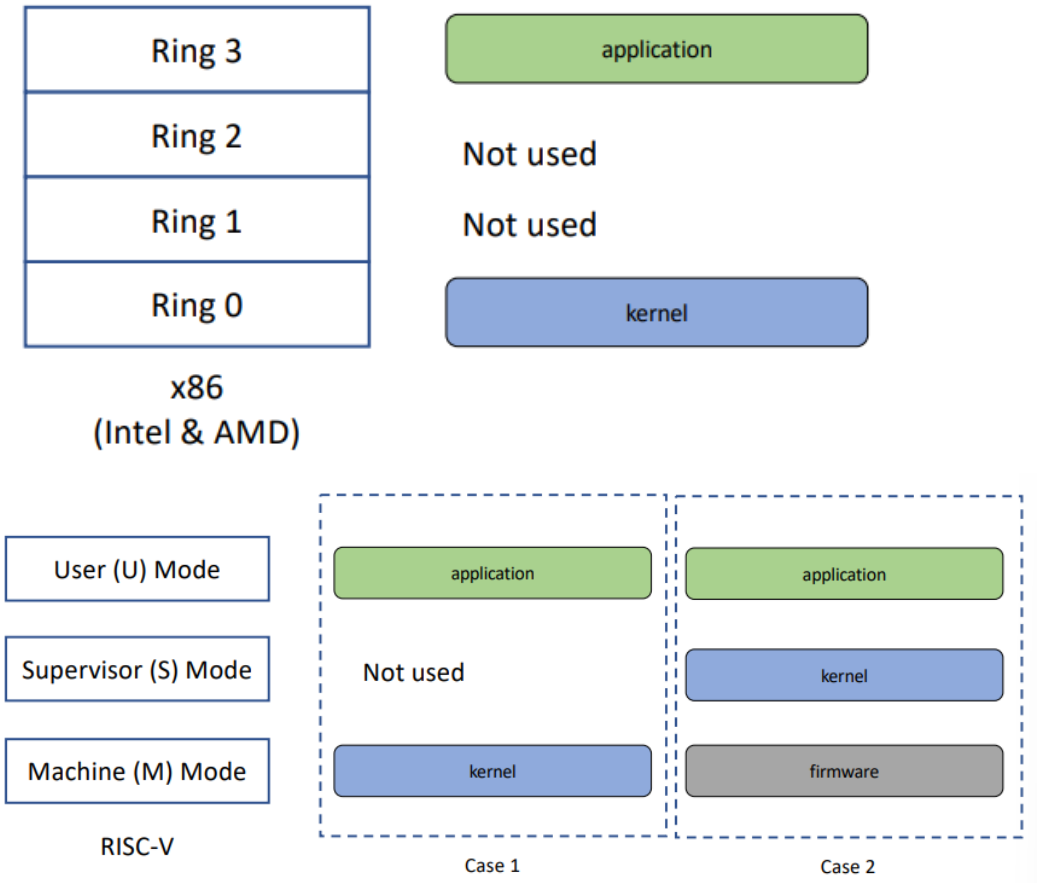
Kernel Mode vs. User Mode



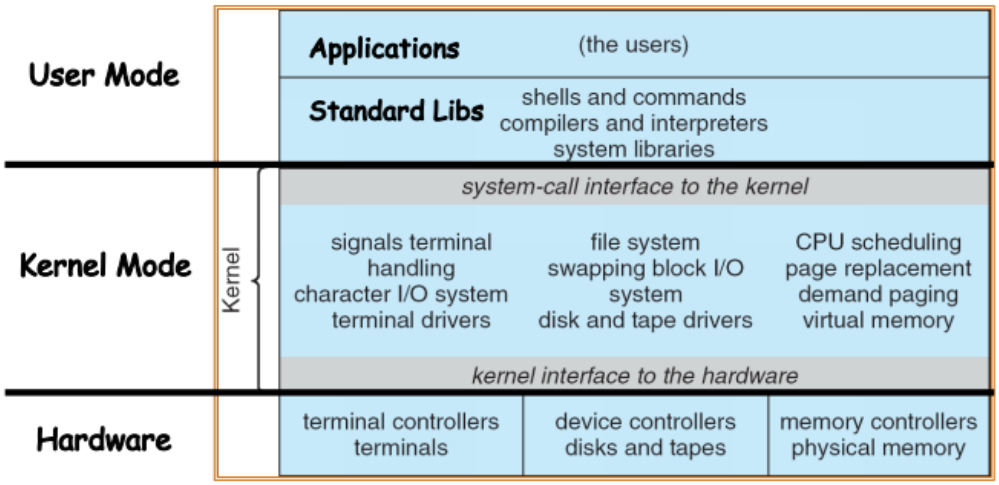
- Hardware provides at least two modes
  - **Kernel mode**: Run kernel code
  - **User mode**: Normal programs executed
- What is needed in the hardware to support "dual mode" operation?
  - A bit of **state** (user / system mode bit), **can not be updated by user mode**
  - Certain **operations / actions** only permitted in system / kernel mode
    - In user mode they **fail** or **trap**
  - User → Kernel transition **sets** system mode AND saves the user PC (program counter)
    - Operating system code carefully puts aside user state then performs the necessary operations
  - Kernel → User transition **clears** system mode AND restores appropriate user PC

- Dual-mode operation allows OS to **protect itself** and other system components
  - Mode **bits** provided by **CPU hardware**
  - Provides ability to **distinguish** when system is running user code or kernel code
  - Some instructions designated as **privileged**, only **executable in kernel mode**

Mode Bits in CPUs

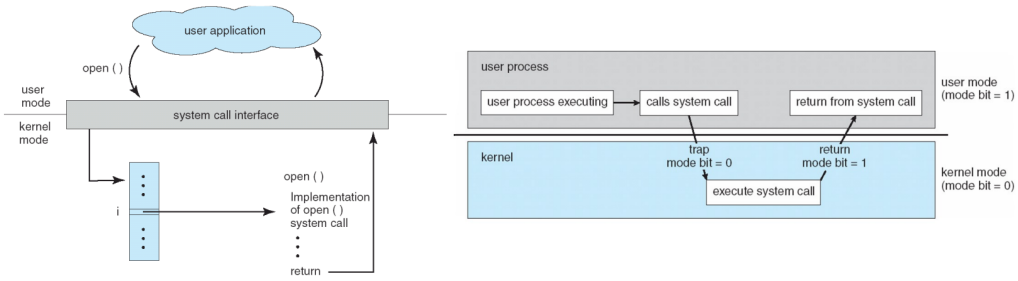


Unix System Structure



2. Three types Mode Transitions Kernel → User

System Calls



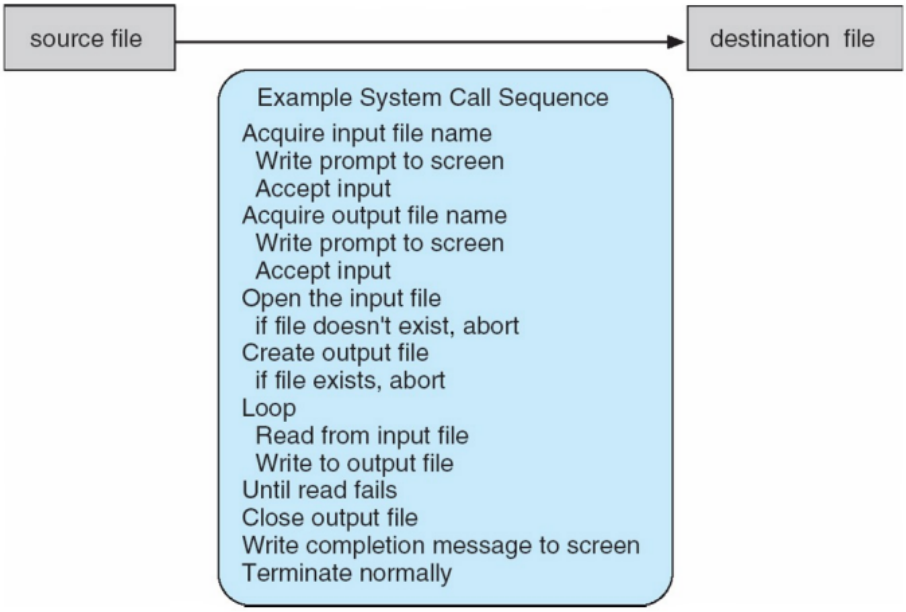
- Programming interface to the services provided by the OS
- Typically written in a high-level language (C or C++)
- API, provided for user mode to call
    - Process requests a system service, e.g., exit
    - Like a function call, but “outside” the process
    - Does not have the address of the system function to call
    - Marshall the syscall id and args in registers and exec syscall

Implementation

- Typically, **a number** associated with each system call
    - System-call interface maintains a **table indexed** according to these numbers
- The system call interface invokes intended system call in OS kernel and **returns status** of the system call and any **return values**
- The caller needs to know **nothing** about how the system call is implemented
    - Just needs to obey calling convention and understand what OS will do
    - Most details of OS interface **hidden** from programmer by library API

Example of System Calls

Copy the contents of one file to another file



Types of System Calls

	Windows	Unix
Process Control	CreateProcess() ExitProcess() WaitForSingleObject()	fork() exit() wait()
File Manipulation	CreateFile() ReadFile() WriteFile() CloseHandle()	open() read() write() close()
Device Manipulation	SetConsoleMode() ReadConsole() WriteConsole()	ioctl() read() write()
Information Maintenance	GetCurrentProcessID() SetTimer() Sleep()	getpid() alarm() sleep()
Communication	CreatePipe() CreateFileMapping() MapViewOfFile()	pipe() shmget() mmap()
Protection	SetFileSecurity() InitlializeSecurityDescriptor() SetSecurityDescriptorGroup()	chmod() umask() chown()

Exception and Interrupt

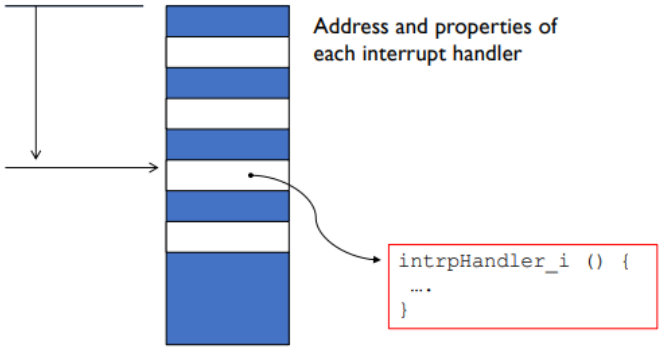
- Exceptions (**synchronous**) react to an abnormal condition
  - eg. Divide by zero, Illegal memory accesses, Map the swapped-out page back to memory
- Interrupts (**asynchronous**) preempt(先占，先取) normal execution
  - Notification from device (e.g., new packets, disk **I/O** completed)

- Preemptive scheduling (e.g., **timer** ticks)
- Notification from another CPU (i.e., Inter-processor Interrupts)
- Independent of user process

Attention

- Interrupt and Exception are handled in **the same way**
- If you run the same program, **exception will always appear**, while **interrupt will not always appear**

Procedure



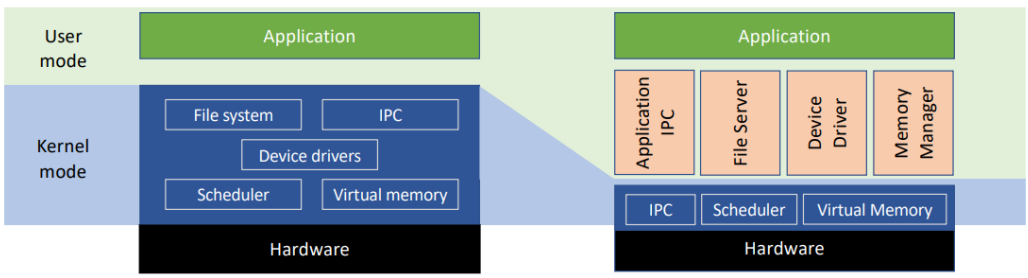
- Stop execution of the current program
- Start execution of a handler
- Processor accesses the handler through an entry in the **Interrupt Descriptor Table (IDT)** (as the figure shows)
- Each interrupt is defined by a number

3. Kernel Structures

Monolithic Kernel

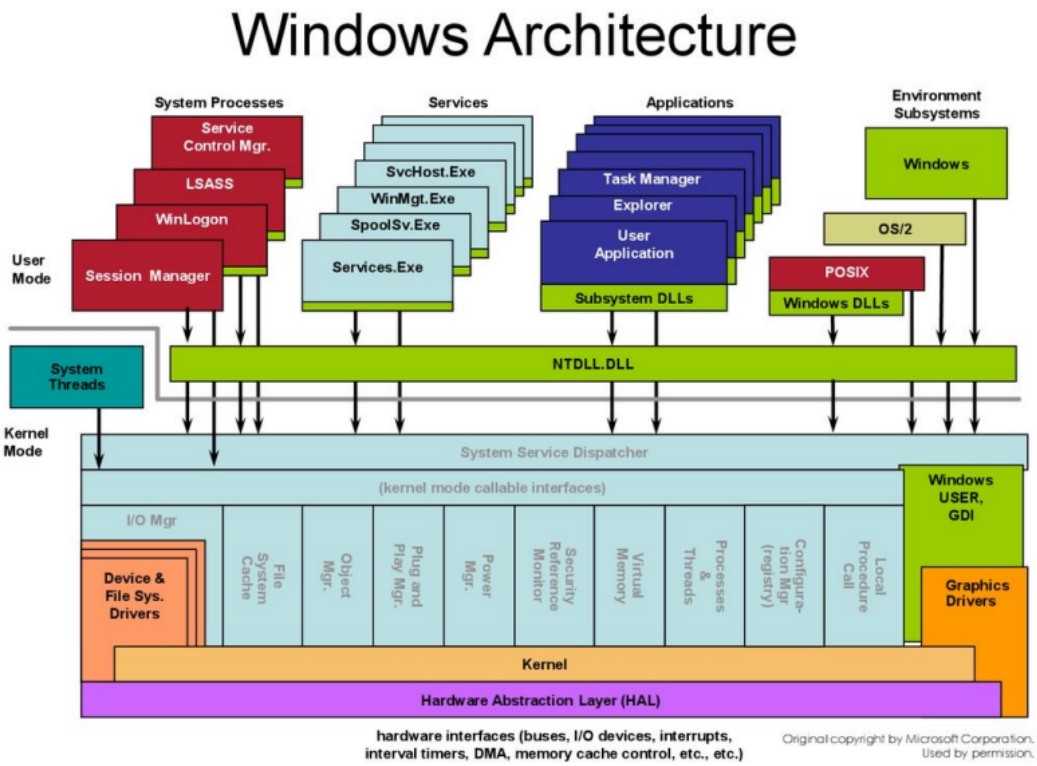
- A monolithic(大一统) kernel is an operating system software framework that holds all privileges to access I/O devices, memory, hardware interrupts and the CPU stack
- Monolithic kernels contain many components, such as memory subsystems and I/O subsystems, and are usually **very large**
  - Including filesystems, device drivers, etc.
- Monolithic kernel is the basis for Linux, Unix, MS-DOS

Micro Kernel



- Microkernels outsource the traditional operating system functionality to ordinary user processes for better flexibility, security, and fault tolerance
- OS functionalities are pushed to **user-level servers** (e.g., user-level memory manager)
- User-level servers are trusted by the kernel (often run as root)
- Protection mechanisms stay in kernel while resource management policies go to the user-level servers
- Pros
    - Kernel is more responsive (kernel functions in preemptible user-level processes)
    - Better stability and security (less code in kernel)
    - Better support of concurrency and distributed OS (later…..)
- Cons
    - More IPC needed and thus more context switches (slower)

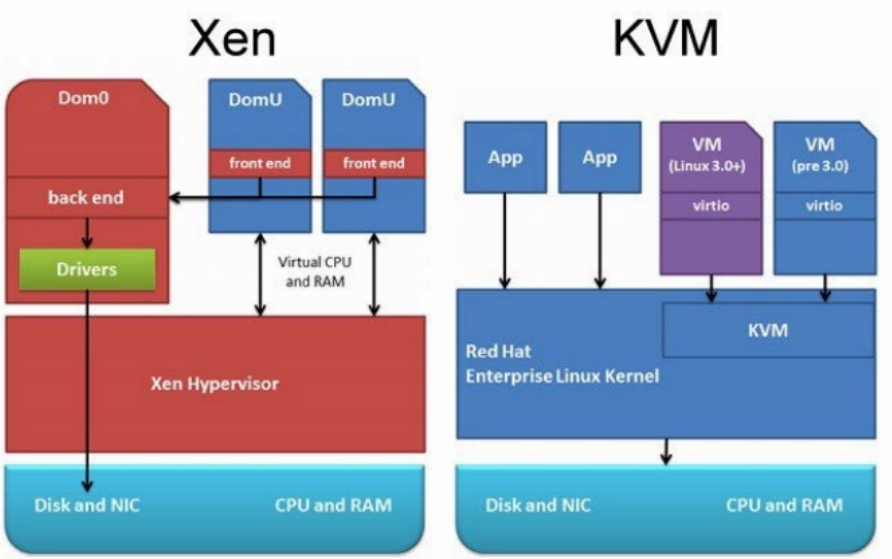
Hybrid Kernel



- A combination of a monolithic kernel and a micro kernel
    - Example: Windows OS

Virtualization and Hypervisors

Hypervisor (or virtual machine manager/monitor, or VMM) emphasizes on **virtualization** and **isolation**



- OS can run on hypervisor (almost) without modification
- Resource partition among VMs
- Micro kernel and exokernel can sometimes be used to implement hypervisors

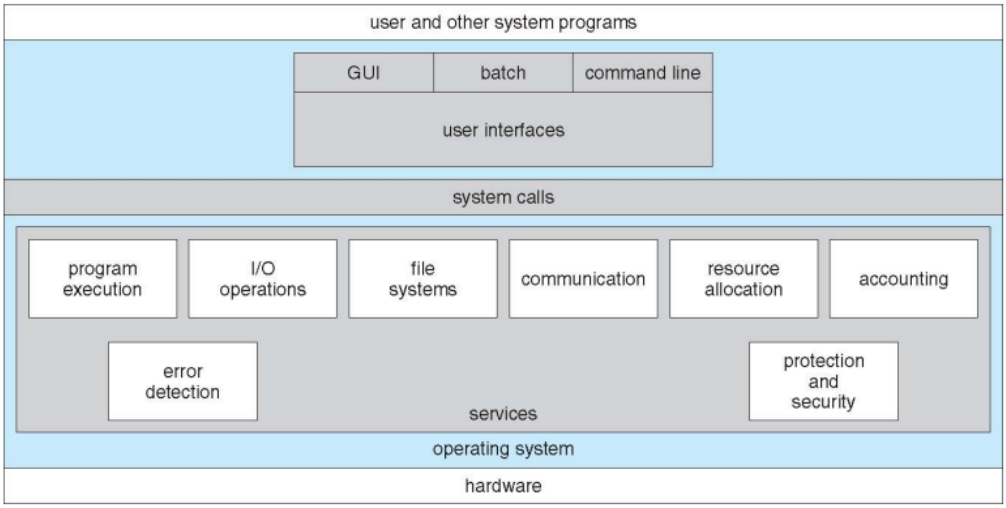
4. OS Design and Service

OS Design Principles

- Internal structure of different Operating Systems can vary widely
  - Start by defining goals and specifications
  - Affected by choice of hardware, type of system
- User goals and System goals
  - **User goals**: operating system should be convenient to use, easy to learn, reliable, safe, and fast
  - **System goals**: operating system should be easy to design, implement, and maintain, as well as flexible, reliable, error-free, and efficient
- OS separates **policies** and **mechanisms**
  - **Policy**: which software could access which resource at what time
    - if two processes access the same device at the same time, which one goes first
    - if a process hopes to read from keyboard
  - **Mechanism**: How is the policy enforced
    - request queues for devices, running queues for CPUs
    - access control list for files, devices, etc.
  - The separation of policy from mechanism is a very important principle, it allows maximum flexibility if policy decisions are to be changed later

Operating System Services

Operating system provides a set of services to application programs



Functions that are helpful to the user

- **User interface**: Almost all operating systems have a user interface (UI)
    - Varies between Command-Line (CLI), Graphics User Interface (GUI), Batch
- **Program execution**: The system must be able to load a program into memory and to run that program, end execution, either normally or abnormally (indicating error)
- **I/O operations**: A running program may require I/O, which may involve a file or an I/O device
- **File-system manipulation**: The file system is of particular interest. Obviously, programs need to read and write files and directories, create and delete them, search them, list file Information, permission management
- **Communications**: Processes may exchange information, on the same computer or between computers over a network
    - Communications may be via shared memory or through message passing (packets moved by the OS)
- **Error detection**: OS needs to be constantly aware of possible errors
    - May occur in the CPU and memory hardware, in I/O devices, in user program
    - For each type of error, OS should take the appropriate action to ensure correct and consistent computing
    - Debugging facilities can greatly enhance the user's and programmer's abilities to efficiently use the system

Functions that ensure the efficient operation of the system itself via resource sharing

- **Resource allocation**: When multiple users or multiple jobs running concurrently, resources must be allocated to each of them
    - Many types of resources - Some (such as CPU cycles, main memory, and file storage) may have special allocation code, others (such as I/O devices) may have general request and release code
- **Accounting**: To keep track of which users use how much and what kinds of computer resources
- **Protection and Security**: The owners of information stored in a multiuser or networked computer system may want to control use of that information, concurrent processes should not interfere with each other
    - Protection involves ensuring that all access to system resources is controlled
    - Security of the system from outsiders requires user authentication, extends to defending external I/O devices from invalid access attempts
    - If a system is to be protected and secure, precautions must be instituted throughout it. A chain is only as strong as its weakest link

System Programs

Most users' view of the operation system is defined by system programs

- **File management**
    - Create, delete, copy, rename, print, dump, list, and generally manipulate files and directories
- **Status information**
    - Some ask the system for info - date, time, amount of available memory, disk space, number of users
    - Others provide detailed performance, logging, and debugging information
    - Typically, these programs format and print the output to the terminal or other output devices
    - Some systems implement a registry - used to store and retrieve configuration information
- **File modification**
    - Text editors to create and modify files
    - Special commands to search contents of files or perform transformations of the text
- **Programming-language support**
    - Compilers, assemblers, debuggers and interpreters sometimes provided

- **Program loading and execution**
- **Communications**
    - Provide the mechanism for creating virtual connections among processes, users, and computer systems
    - Allow users to send messages to one another's screens, browse web pages, send electronic-mail messages, log in remotely, transfer files from one machine to another