intro

BASICS OF THE UNIX/LINUX ENVIRONMENT

What is an operating system (OS or O/S)?

- Interface between Hardware and User.
- It is a program (software) designed to manage and coordinate activities and resources of the computer.
- Controls the hardware (physical part of the computer) and other software.
- Controls how other applications (=programs) are implemented.

See: http://en.wikipedia.org/wiki/Operating_system

OS's at CERI

- Solaris 9 UNIX
  - House 3 Sun Lab, Long Building, many offices
  - 2 Graphical Desktop Environment options:
    - Common Desktop Environment (traditional)
    - GNOME 2.0 (more PC-like)
- Mac OS X
  - 1 in Long Building, many faculty offices.
- Windows (XP)
  - Lab in Long Building, many student offices.
- Various flavors of Linux
  - Popular, open source version of UNIX (often described as "UNIX-like") found on a number of machines at CERI, but not officially supported at CERI.

Why learn Unix/Linux?

- Because you have no choice

  ("Resistance is futile", The Borg, Star Trek).

- It is what is running in most geophysics departments.
- Most geophysics tools (SAC, GMT, GAMIT/GLOBK, etc.) only run on Unix (although there is a Windows version of GMT).

(~89% of the worlds computers run some form of Windows, ~10% run some form of the Mac OS, and ~1% run some flavor of Unix.)

Why learn Unix/Linux?

- Designed to be multi-user (from the dark ages when all computers were shared), interactive (as opposed to "batch"), and multi-tasking.
- Invented by and for computer scientists/ system programmers (not users, unfortunately).

Why learn Unix/Linux?

- Powerful, flexible, and small
- Hardware independent

(these two points are much more important to manufacturers and designers than general users, i.e. us)

Why learn Unix/Linux?

- "Free" from Bell Labs and Berkley (this is why is it still around).
- Open source – "free" – applications, including compilers.
  - Most common free applications designed as part of the GNU Project (GNU's Not Unix)
- It is what is running in most geoscience (both university and corporate) labs.

A bit of history – see:

http://www.bell-labs.com/history/unix/

- Originally developed at AT&T in the late 60s/ early 70s.
- Freely given to universities in the 70s.
- Berkeley scientists continued to develop the OS as BSD Unix in parallel with AT&T (AT&T eventually licensed it for commercial use).
- Much development, branching, and combining has led to the most common variants of Unix ("flavors" or "distributions" in Unix speak).

Common flavors

- Solaris 9 Unix
  - Distributed by Sun Microsystems, runs on Sun Hardware, PC hardware.
  - Derived from Unix System V release (AT&T) on a Unix kernel.
- Mac OSX
  - Distributed by Apple, runs on Mac Hardware.
  - Derived from BSD Unix OS on a Mach kernel.
- Linux
  - Free* and commercial# versions available built on a Linux kernel.
  - Flavors most likely to hear about are RedHat#, Ubuntu*, Fedora*, Debian*, Suse*,....

Does this matter?

- No, the differences between the various flavors of the Unix operating system should not severely affect your work in this class or even much of your research at CERI.

BUT

Does this matter?

- Yes, you need to be aware of OS differences
  - When file sharing with others (this is more of a hardware, rather than an OS issue).
  - When compiling source code (the executable file is married to hardware).
  - If sharing programs, shell scripts, etc. with others.
  - Or if moving between the different systems at CERI.

Relation to Windows

None.

Windows XP

Built on MS-DOS (which is not really an operating system), which has nothing to do with Unix and everything to do with Microsoft.

Cygwin – unix/linux like environment for windows. Have to build everything from source.

Relation to Windows

The differences between the Unix Philosophy and the Windows Philosophy … can be boiled down into a question of smarts … .

Unix and Windows store the smarts in different places.

Unix stores the smarts in the user; Windows stores the smarts in the OS.

Learning curves

Enter the concept of the “Learning Curve”. …

A "steep" learning curve generally refers to something that requires a lot of initial learning.

A "shallow" learning curve is exactly the opposite; a very gentle learning means, rather than a requirement for flash-cramming.

Learning curves

Armed with those definitions, it's fairly simple to then go ahead and say that Unix has an inherently steep learning curve, and Windows has a very shallow one.

Windows

Our Microsoft brethren have taken the approach of making the shallowest possible learning curve. To take a cue from the fast food industry, Windows is the "under-3" toy of the OS world. The ultimate goal is to flat-out destroy any barrier to entry by removing any requirement for initial knowledge or learning of how and why, and of making the system simplistic enough that it can be used without any understanding of how it works.

Unix

The Unix crowd has taken the opposite approach. Unix has a steep learning curve; it doesn't shield the user from complexity; rather, it revels in the complexity. It recognizes that a general-purpose computer is a fiendishly complicated device capable of doing an unbelievable assortment of things.

Unix

It recognizes that the computer is a tool of the user, and so takes a tool-building philosophy. Make a lot of tools, and make each tool specific, and let the user select the tool they think appropriate, and let the user combine the tools however they want. It's not aimed at making things easy; it's aimed at making things possible.

User ID and passwords

BASICS OF THE UNIX/LINUX ENVIRONMENT

User ID and Passwords

- User ID: usually a derivative of your name and the same as the beginning of your email.
  - Your CERI and UoM user ids are the same.

(User ID's on the UM system can only be a maximum of 8 characters due to the limitations of early computes. Unix is full of such anachronisms.

User ID's are formed using an algorithm. Take first initial (and maybe middle) and full last name, if that is more than 8 characters, start removing vowels from the back, if still longer than 8 characters, start removing consonants from the back.)

User ID and Passwords

- Password: a (hopefully complicated, hard to guess [and therefore remember]) combination of upper and lower case characters, symbols, and numbers that allows access to the account.
  - Your CERI Unix password is unique to the Sun systems.
  - Your UoM Outlook email, Spectrum, Tiger labs, and CERI PC lab password is the same for all these systems (because they all access your primary UoM account).
  - If you need/use the non-Sun systems (Unix machines in the GPS lab, a faculty member's MAC, etc.) you will have a unique username/password for each of them.

Passwords

- Do not tell anyone your password!!!!
- Do not leave your password sitting around on a post-it note.
- Do not email your password.

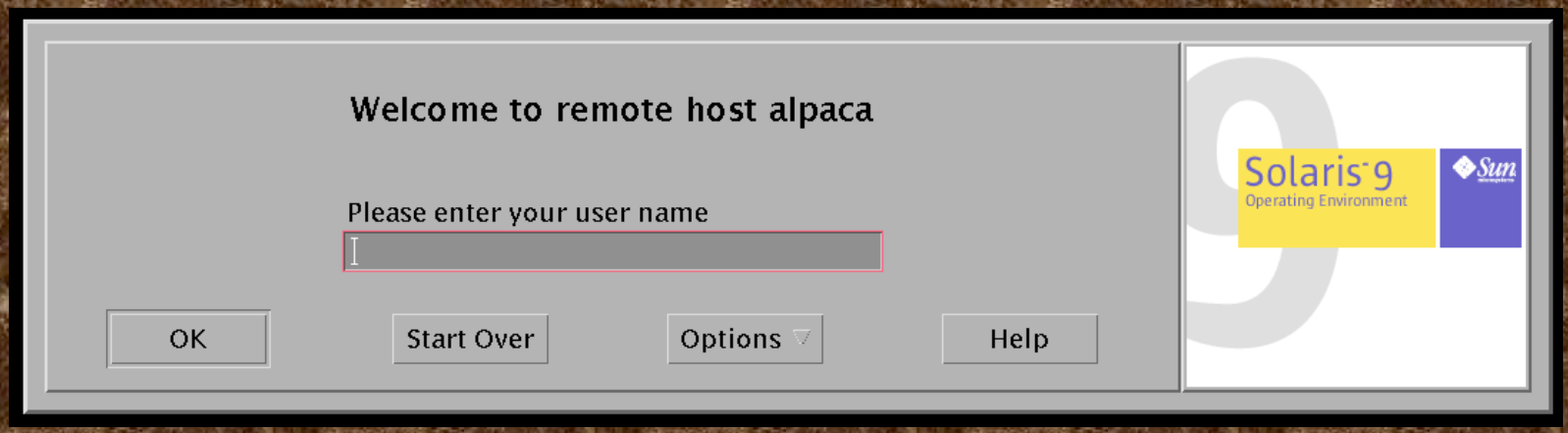
- If you forget your password, you have to visit the system administrator and (humbly) ask for a new one. There is no way for anyone (except hackers) to figure it out.

Accessing unix from Sun lab

- Sit down.
- If the screen is dark – hit any key (the shift key is the "safest" as it does not actually send anything to the comptuer) or move the mouse to "wake it up".

Accessing unix from Sun lab

- You will see something like the following.

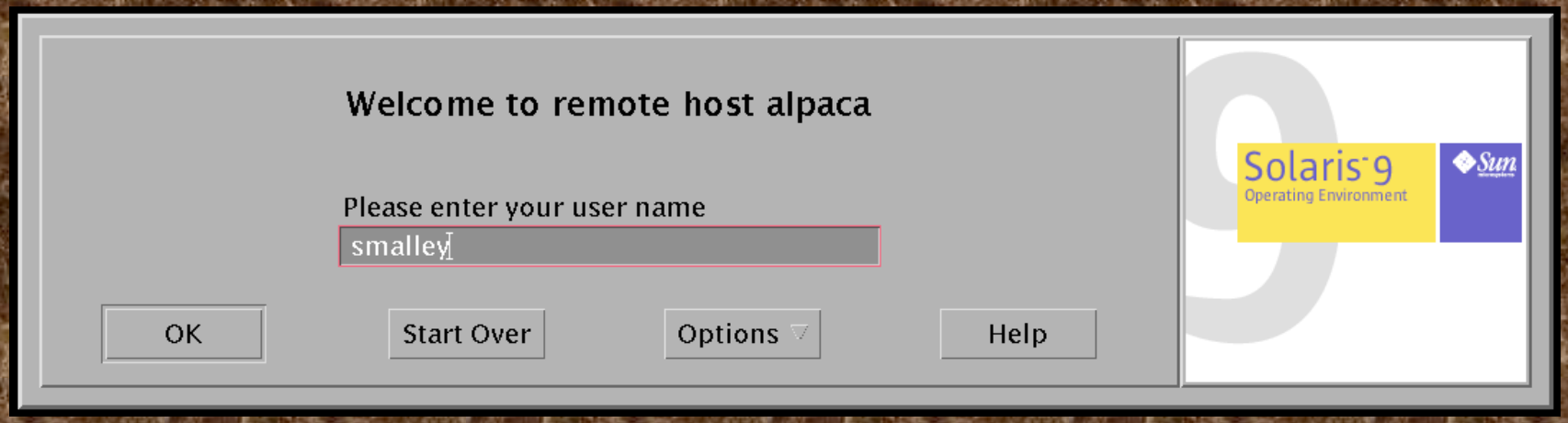


The "host" is the name of the computer you are connecting to.

Note the "cursor" (flashing "I" beam thingy, this is where your typing will go).

Accessing unix from Sun lab

- Type your CERI/UoM user id (uuid) in the user id field.



You can also put the mouse over the boxes and click to do various other things.

Accessing unix from Sun lab

- Hit "Enter" (also known as a "Carriage Return" or <CR> from the dark ages of typewriters), or click the "OK" box, to get.

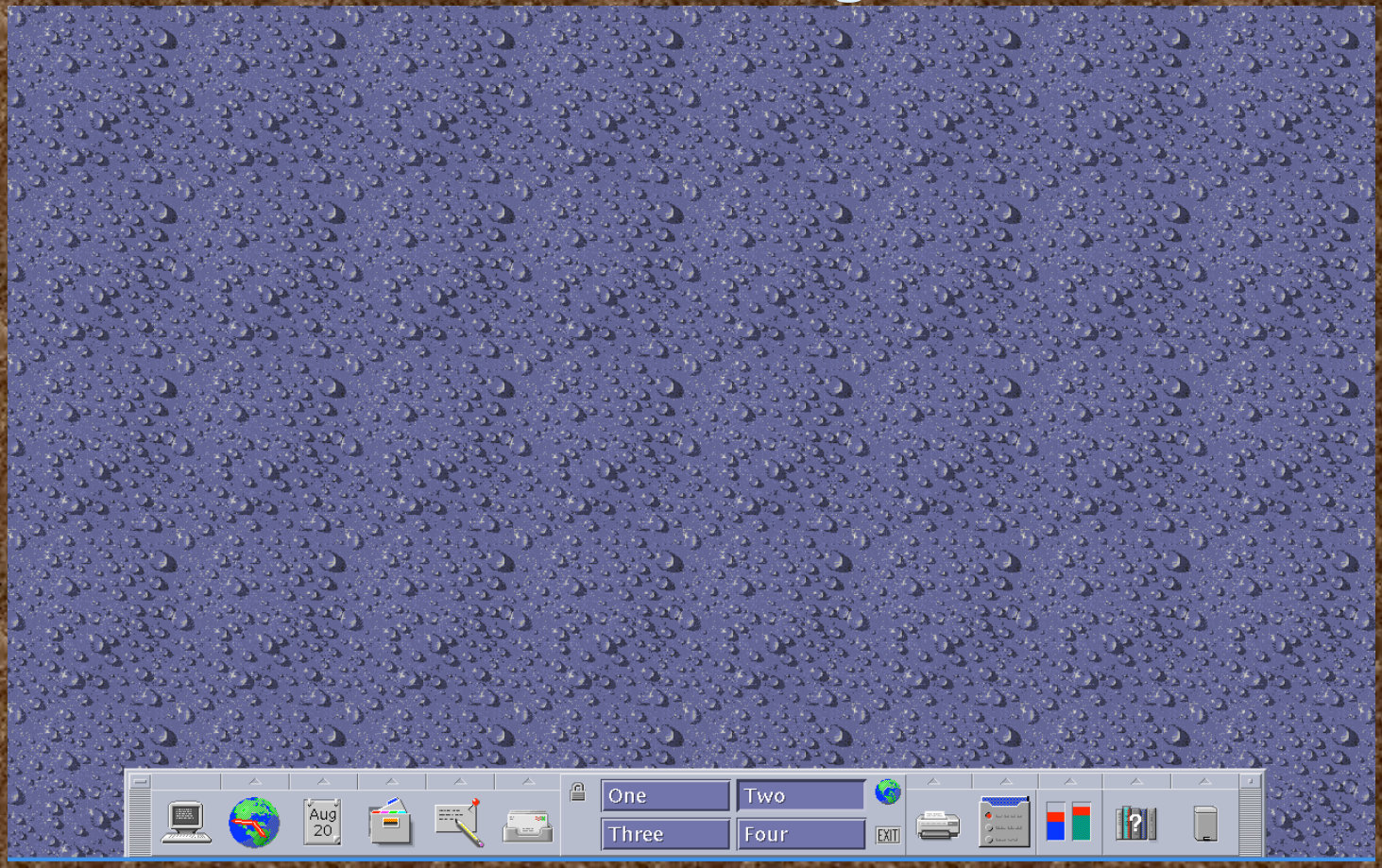
Accessing unix from Sun lab

- Enter your password in the password field.

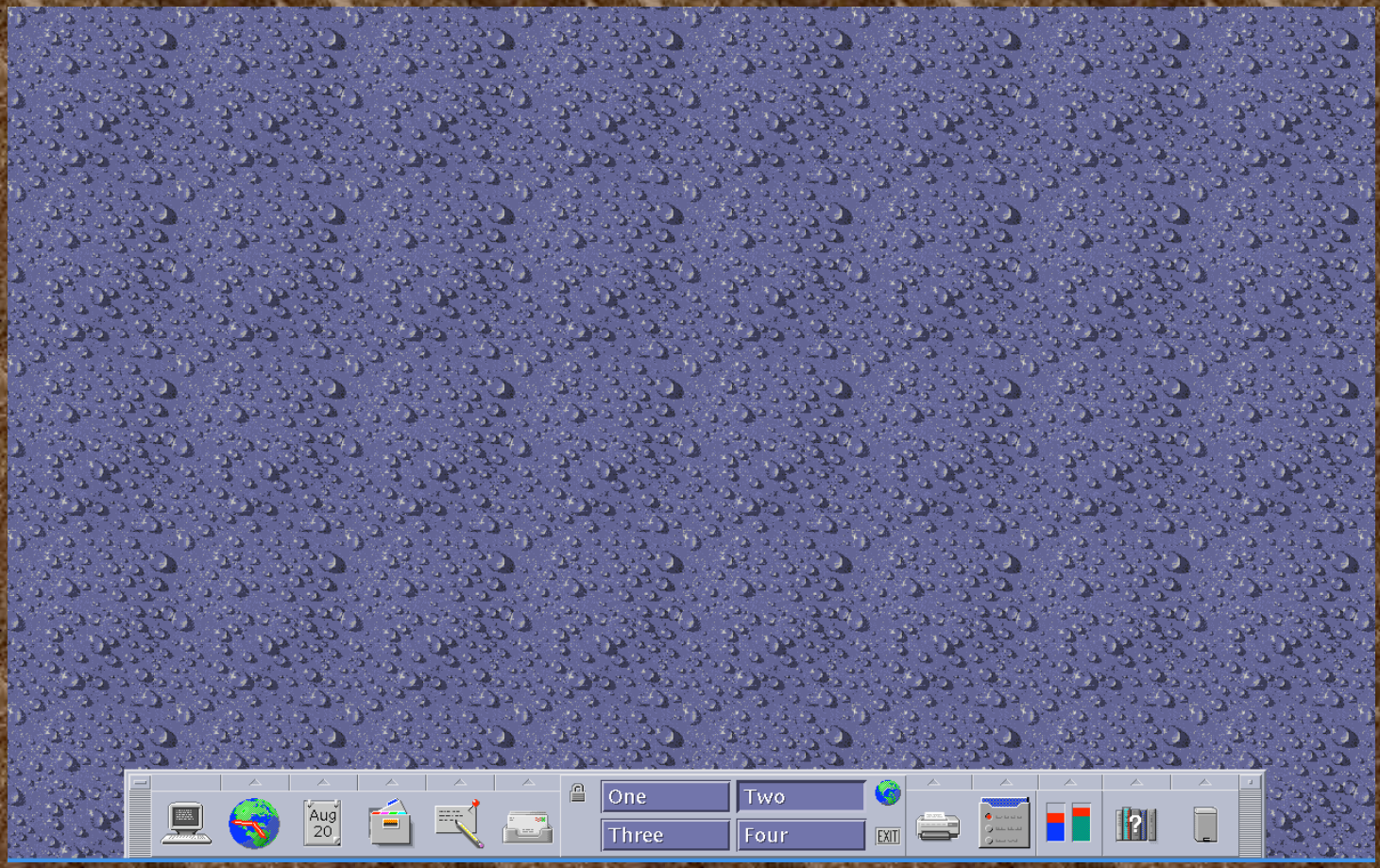
You will not see anything as you type.

Note that Unix machines are "case sensitive" meaning they distinguish between upper and lower case letters. "Bob" and "bob" are two different entries.

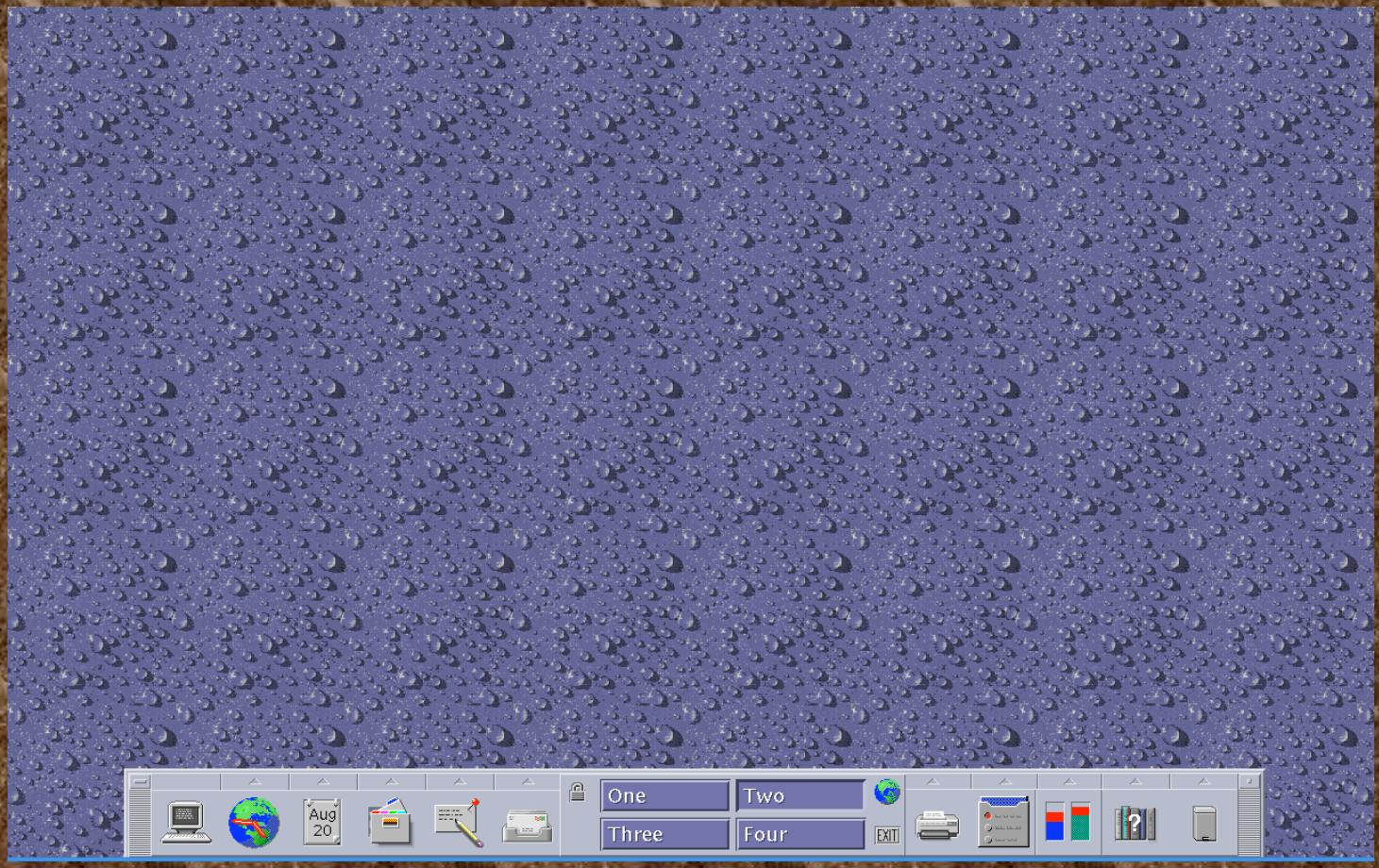
You will now see something like this.



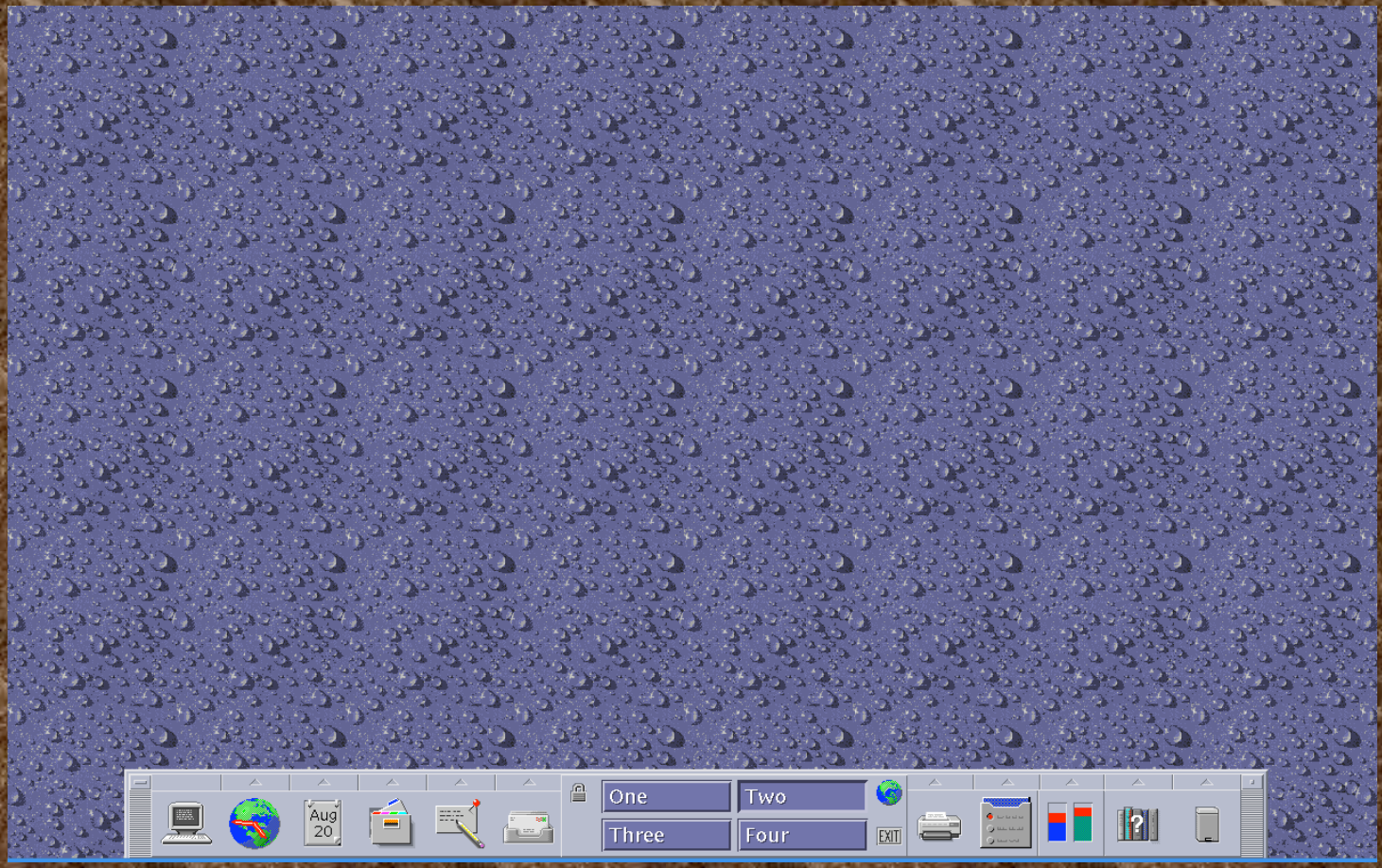
From here – starting at the globe icon and going right - you can surf the web, manage a calendar, look in your folders, open a text editor, do email. Place the mouse over an icon for a few seconds to see what it does.



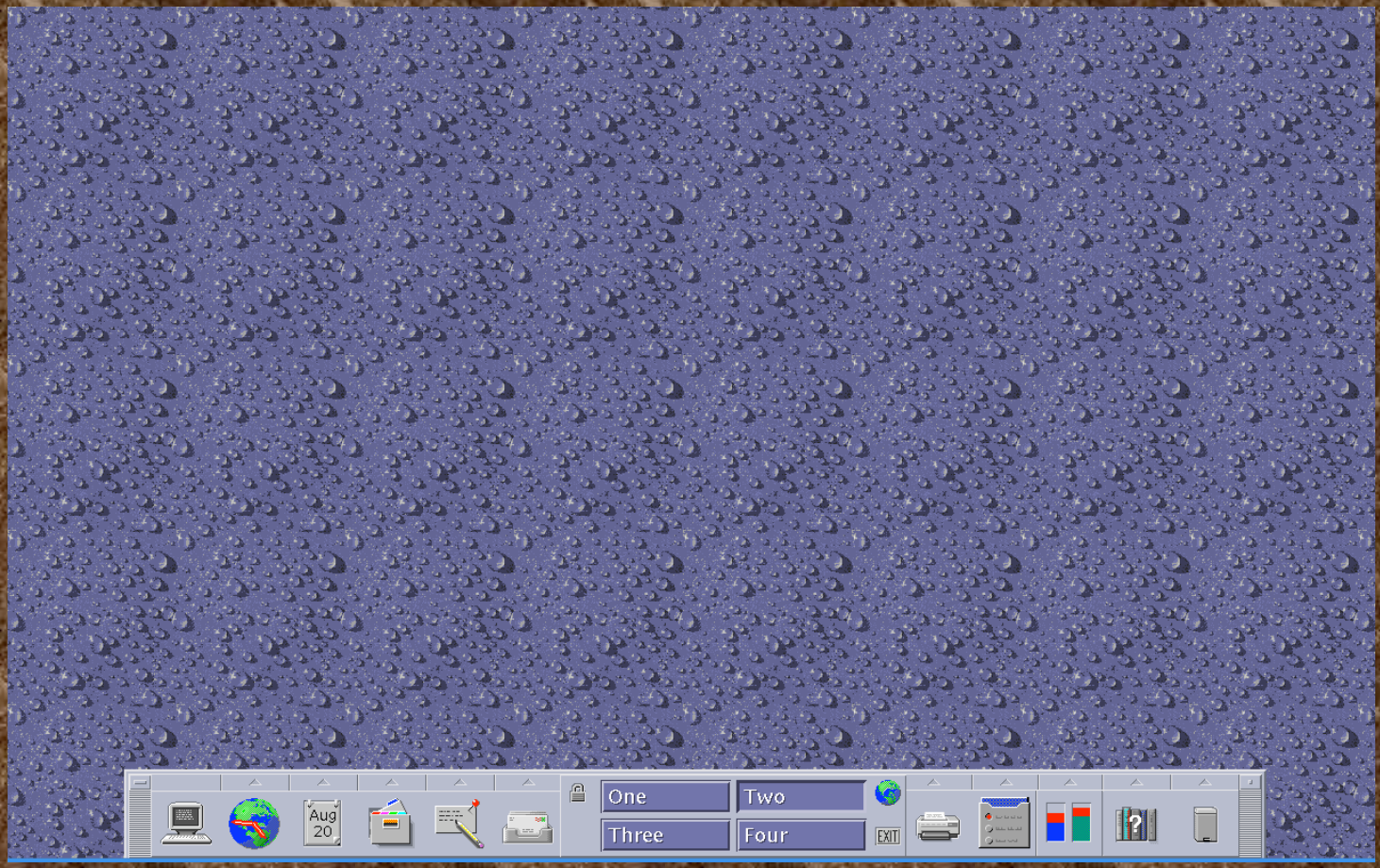
The buttons in the middle labeled "One" through "Four" cycle through 4 desktops. (We are in desktop #2, whose "button" looks like it is "pushed-in")



For the time being - forget about the three little icons in the center box and the ones on the right except the rightmost one, which is the trash.



In order to do something useful, we have to have a way to tell the computer what we want to do.



To do this we need to open a terminal window, which is done by clicking on the terminal icon in the lower left. (note the icons may be in different orders, and there may be a different selection of them.)

We have now entered the world of the "command line", which is where most of your interaction with the Unix system will occur.

Let's do our first useful thing: change your Password to something you can remember.

(you must do this on your first login.)

- Unix passwords can be easily changed using the *passwd* command

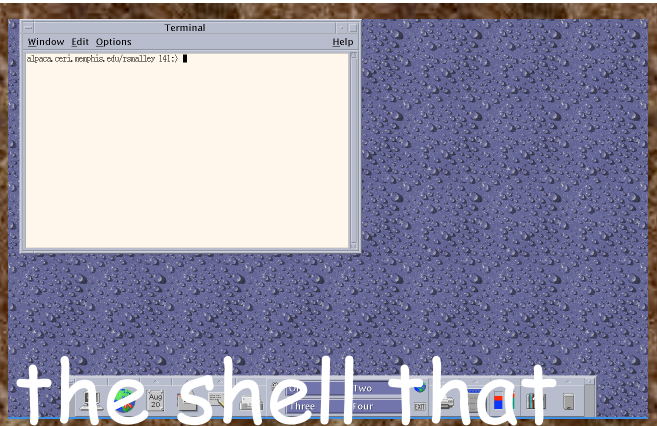
Note two things:

1) It will be easy
2) It uses an intuitively obvious "command", *passwd*

These two observations will apply to almost all of our interactions with UNIX and serve as an introduction to the Unix philosophy (which will be presented shortly).

passwd command

%passwd -r nis <CR>



The % is the "prompt" from the shell that says it is ready read input. There is also usually a flashing cursor after the prompt.

Commands we will enter are shown in italics.

<CR> is Carriage Return/Enter.

This activity occurs in the terminal window we previously opened.

This will start a little dialog with you. Answer the requests for your Old Password, your New Password and its confirmation as they come up.

%passwd -r nis<CR>

Changing password for your_username

Old password: *type old password<CR>*

New password: *type new password<CR>*

Re-type new password: *type new password again<CR>*

What not to use for your password:

- Your name, address, phone number, or anything that is common knowledge, or can be easily guessed, about you.

- Common words.

Password guessing programs don't get tired and can try 1000's of permutations of passwords based on the above.

What to use for your password:

- Something you will not forget.
- Mix in numbers and special characters (1, 2, 3 , !, @, #, $, %, etc.).
- Should be at least 6 characters long (some systems require longer ones).
- (don't use items from "do not use" list with "1" substituted for "i", "0" substituted or "o", etc., the password guessers will try these.)

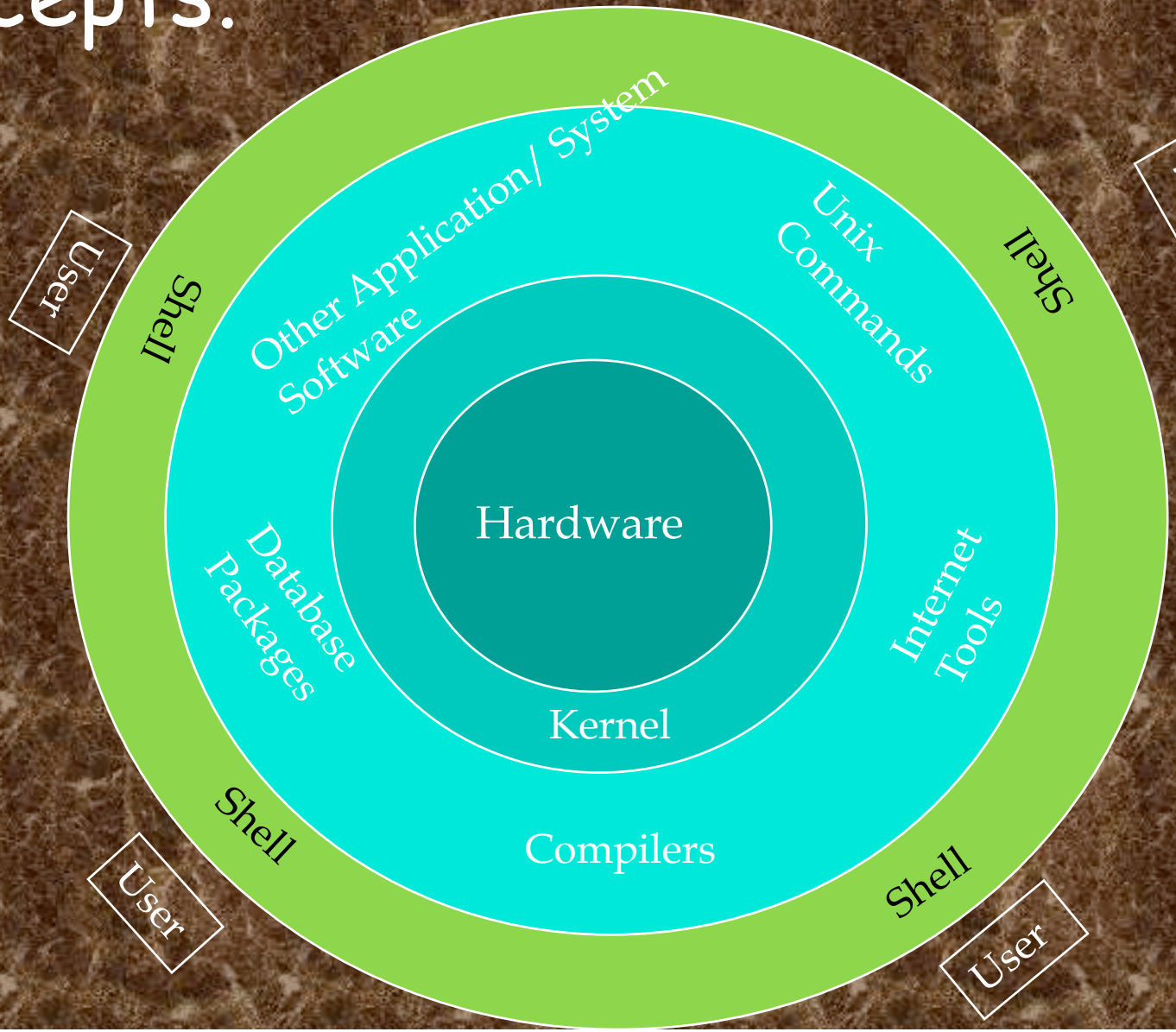
Unix Philosophy

BASICS OF THE UNIX/LINUX ENVIRONMENT

(Mac) (Unix)

"Dilbert" by, Scott Adams, Sep 30, 1994.

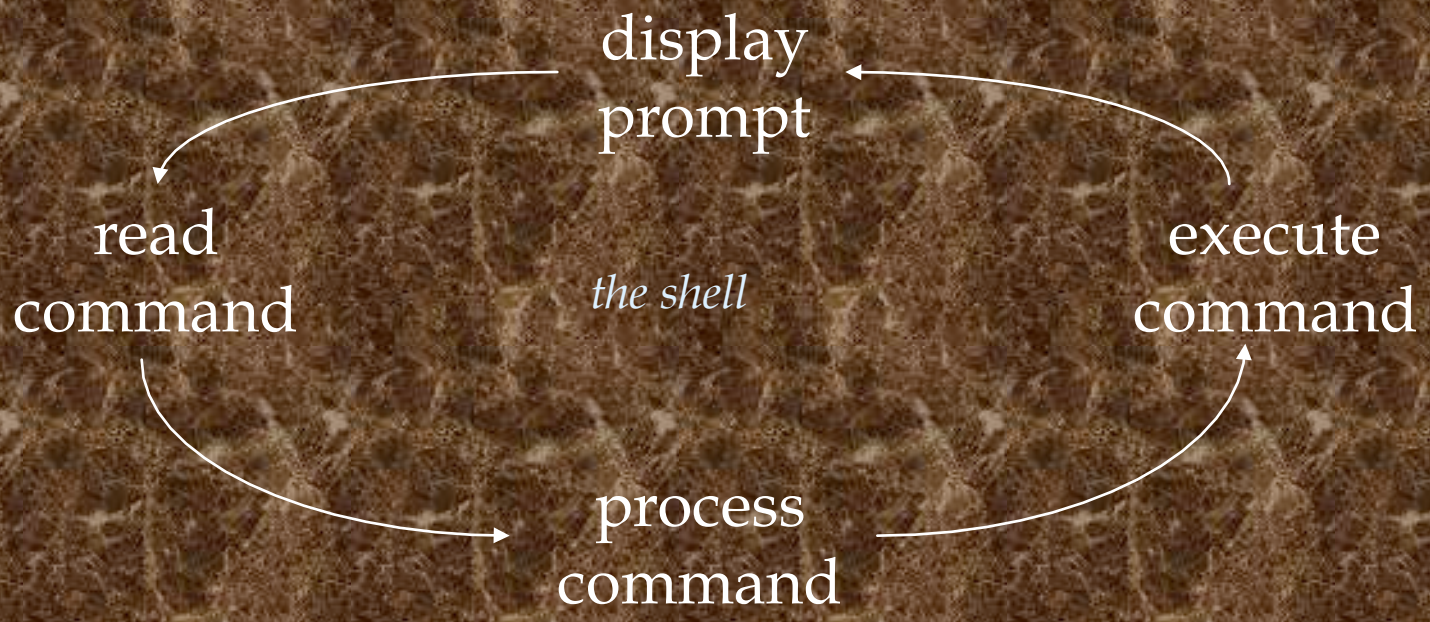
Backing up a bit to illustrate some concepts.



- Hardware – the physical computer.
- Kernel – program, usually hardware dependent, that runs the core or key components of the operating system (process, memory, file, device, and network management).
- Programs/Applications – hardware independent – unix commands, compilers, applications
- Shell – hardware independent - how the user interacts with the Programs/Applications layer.

The Shell

- The UNIX user interface is called the *shell*.
- The shell does 4 jobs repeatedly:



Final Model


Understanding Computer Technology
Main storage
Floppy disk
Application software
Function key
Users interface
Surge control device
input
Central processing unit
Backup system
Overflow (input / output error)
De-bugging tool
Peripherals (hardware)
Supplementary data

We will now take a short detour to examine the Unix philosophy.

It will keep returning to haunt us, but if you understand it, it will make the process less painful.

What is the "Unix Philosophy"?

(can computer operating systems have a "philosophy"?)

According to Doug McIlroy

(i)Make each program do one thing well.

So, to do a new job, build afresh rather than complicate old programs by adding new features (otherwise known as "bells and whistles").

What is "Unix Philosophy"?

Machine shop vs. appliance
(gives you the tools and you to make appliance)

What is "Unix Philosophy"?

Advantage

- POWERFUL

What is "Unix Philosophy"?

Disadvantages

- Lots of reinventing the wheel
- Requires more educated user
- Requires more work from user rather than developer

What is "Unix Philosophy"?

Typical question: can UNIX do this?

Typical answer: NO, but YOU can write a program!

Unix enthusiasts think this is the answer the average user wants to hear!

"UNIX Philosophy"

(ii) Expect the output of every program to become the input to another, as yet unknown, program.

- Don't clutter the output with extraneous information.

"UNIX Philosophy"

Unfortunately this may make things confusing for the uninitiated user.

The output is for "next program" in a pipe, not the user.

"UNIX Philosophy"

Idea of "filter" -

Every program takes its input from Standard IN (originally a teletype, now a keyboard),

does something to it ("filters" it) and

sends it to Standard OUT (originally a teletype, now a screen)

(notice that the "user" is not part of this model).

"UNIX Philosophy"

Idea/use of – redirection ("<", "<<" and ">", ">>")

- Take input from a file rather than Standard IN

- Send output to a file rather than Standard OUT

(Unix treats everything like a "file")

"UNIX Philosophy"

Idea/use of - pipes ("|")

Take input from previous program

Send output to next program

"UNIX Philosophy"

Idea/use of – command substitution (` …`)

Use the output of a command as 'some sort of input' to another command.

"UNIX Philosophy"

Unix put lots of single minded programs in a row (with pipes) to do what you need.

(Don't use temporary/intermediate files – use pipe).

"UNIX Philosophy" Continued

Avoid stringently columnar or binary input formats.

(Avoid, but sometimes necessary. Not closely followed by many programs.)

Don't insist on interactive input.

(Does not fit in with use of pipes.)

Instead, control is implemented by use of "command line switches"

Write programs to handle text streams, because that is a universal interface.

(fine if you're a system programmer, not always so useful for scientific data crunching.)

REVIEW

Write programs that do one thing and do it well.

(lean and mean)

Write programs to work together.

(pipes)