

No lecture on Friday

Exhibition day for Stage IV engineers.

Stage III engineers should attend.

Where: SE lab 1:30 - 4:30

Stage III Computer Science students have a research group presentation.

If you are considering doing Hons or PostGrad Diploma in Science you should attend.

Where: Library B15

When: 2pm (340 lecture room and time)

OS User-interface

What do we want?

To produce results as instructed.

The desired results are almost always produced by user-level programs. The role of the operating system includes:

- starting these programs up
- handling the input and output requirements of the programs
- coping with errors which occur as the programs run
- indicating to the user (or something else) whether the program completed satisfactorily

The major OS UI task concerns manipulating files.

Telling the system what to do

We need (or want)

- a way of conveying our instructions to the system
- some sort of consistent approach to providing the services
- a way to automate a sequence of commands
- it should be easy to use - in particular common tasks should be easily accessible and simple

Approaches

CLI - **command line interface**, typing a complete command and then pushing "return" or "enter"

Text Menu systems - presenting information and having some way of selecting from the options.

GUI - **graphical user interface**, selection again, but not just from menus.

CLI

e.g. MS-DOS, UNIX

```
Linux
Network Trash Folder
News
Notes for markers.rtf
You have new mail in /usr/spool/mail/robert-s
bash-2.00$ cd 340
bash: cd: 340: No such file or directory
bash-2.00$ cd 415.340/1999
bash-2.00$ ls
a1      local web
bash-2.00$
```

Advantages

- simple to implement
- command interpreter sees first sequence of letters as the name of a program
- the next sequence may be passed as a parameter to that program when it runs
- easy to turn into command files (e.g. shell scripts)
- easy to add flexibility - options can be added as information either for the command interpreter or the program itself. If the interpreter searches a path for command programs then adding a new command is trivial.
- easy for a person with disabilities to use

Knowing the right words

Disadvantages of a CLI

- memorization is required
- semantically incorrect - actually saying run a program when we mean "edit a letter"

UNIX has a hard-won reputation as the best of the worst.

- pwd
- mv
- grep
- cat
- du
- wc

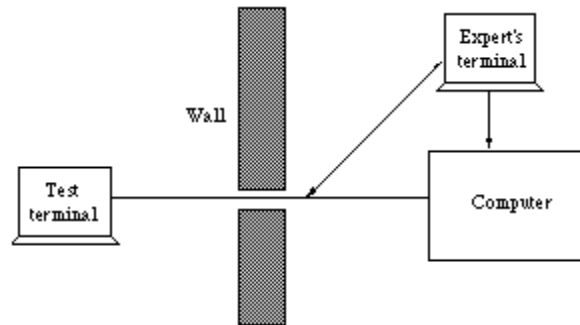
Even commands which sort of do what they say, such as `find` are not simple to use. e.g. to find a file somewhere under the current directory called `A1.class` needs

```
find . -name A1.class
```

and in previous versions of the command needed

```
find . -name A1.class -print
```

Wizard of Oz interface



A method for finding the best words

- the expert user decided if the attempts at instructions were automatically parsable
- if so then typed the correct instruction to the computer
- otherwise sent back an error message
- original system 7% of instructions tried were acceptable to the system
- after analysing the attempts of the users, changes were made and 76% of instructions were then accepted

Text Menu systems

e.g. Transaction processing systems, UCSD
Pascal system

The command level menu looked like this:

COMMAND: E(DIT, R(UN, F(ILE, C(OMP, L(INK, X(ECUTE, A(SSEM, D(EBUG, ?

Advantages

- no need to remember the command name
- brief - single letter or number can start the action
- can be turned into command files (but they look strange - EAQ filename BD)
- easy for people with disabilities

Disadvantages

- what if "edit" is not on the menu?
- less simple to implement
- system needs to know which menu is showing and possibly position of the cursor
- limited flexibility (more screens of menu options), new commands may require recompiling the system.

Implementing menu systems

The UI-system needs a little information about what is displayed on the screen.

- What screen full is shown.

- What (if any) is selected.

Responses which don't match any screen possibilities are rejected.

- What sort of feedback should be given?

Need forms of navigation between screens

- Screens are commonly nodes in a tree

- Up, down, forwards, backwards, main (or home)

- Some of these systems are very sophisticated

 - e.g. info

GUIs

Graphical User Interfaces

e.g. X-windows, Macintosh, Windows 9X – XP

Also has menus but

things on the screen now represent data structures,
programs or objects

coupled with direct manipulation of these objects this
gave an entirely different way of controlling computer
systems



GUIs (cont.)

Advantages

- no need to remember the command name
- much easier to work with multiple programs at the same time
- the system associates an application with every file type (or equivalent)
- icon represents the object and selecting it means I want to work with this object
- is it quicker than a CLI?

Disadvantages

- what if you don't want to do the standard thing? e.g. edit a letter
- what if you don't own the associated program?
- learning what the different icons represent
- how do you start editing a new letter?
- more difficult to use if visually or motor-impaired
- how do you make command files?

WImps – Windows and Icons

Areas of the display.

- associated with a process (or for icons – program, file, directory, rubbish, network, printer, ...)
- the program that displays folders/directories is regarded as part of the OS



What the system needs to know about windows and icons

- what process is in control of this window
- window identifier (possibly more than one window per process)
- position, layer, size (bitmaps for icons)
- can the window be moved / resized / iconified?
- which is the active window (has the focus)?
- is the window a source of drag items
- what type of information can the window accept for drops
- what type of events is the window able to respond to
- what is underneath the window

wiMPs – Menus and Pointers

Menus

- process knows what is possible at this time and presents the possibilities in lists – the list changes as the state changes
- should the UI system be in charge of menus or should that be left up to the programs?
- should menubars be at the top of the screen or at the top of the windows?

What the system needs to know about menus

- list of menu items.
- actions associated with the menu items.
- active items, selected item
- what is underneath the menu

Pointers

- representation on the display of the position (not always visible, e.g., PDA/tablet uses the actual stylus) of a pointing device: mouse, drawing tablet, touch screen...
- used to select windows, icons, menus and menu items and for dragging

What the system needs to know about pointers

- where it is
- state of the buttons of the associated device

The X Protocol

Not a full UI system but a graphical system
various GUIs are built on top of.

Client-Server Design

The design of the X Protocol specifies a client-server relationship between an application and its display.

The X Server

Runs on the local machine.

- manages the keyboard, mouse and display device
- displays drawing requests on the screen
- replies to information requests
- reports an error in a request
- multiplexes keyboard and mouse input onto the network (or via local IPC) to the respective X clients
- creates, maps and destroys windows
- writes and draws in windows

X (cont.)

The X Client

Essentially an application written with the aid of libraries (i.e. Xlib, Xt) that take advantage of the X Protocol.

- displays on an X server
- sends requests to the server
 - e.g. drawing or request for information
- receives events from server
- receives errors from the server

Protocol Messages

Requests

X clients make requests to the X server for a certain action to take place. i.e. Create Window

Replies

The X server will respond to certain X client requests that require a reply.

Events

The X server will forward to the X client an event that the application is expecting. This could include keyboard or mouse input. To minimize network traffic, only expected events are sent to X clients. e.g. Don't want to send all mouse moves.

Errors

The X server will report errors in requests to the X client. Errors are like an event but are handled differently. They are sent to the error handling routine of the X client.

X server design

Device Dependent Layer

Depends on hardware and OS.

Swaps bytes of data from machines with differing byte ordering.

Environment Architectures

Single Threaded Architecture - The X server is a single sequential process using the native time-slice architecture for scheduling demultiplexing requests and multiplexing replies, events and errors among X clients.

Multithreaded Architecture - The X server is a multithreaded process capable of exploiting the nature of the operating system by breaking jobs into multiple threads for the operating system and hardware to perform.

View Apple/Switch ads and spoof

Before next time

Read textbook

3.1.8 Command-Interpreter System