

THE USER VIEW OF OPERATING SYSTEMS (Chapter 16)

Read chapter 16.

We will discuss two important aspects of the OS that pertain to the user:

- (1) the **OS services** that are provided to the user and user's programs
- (2) the medium of delivery of those services, namely the type and appearance of the **user interface** that the system provides and how those services are accessed?

The OS services and the user interface are commonly treated as a **shell**. The user communicates with the OS through a shell. Different shells have their own services, capabilities, and work styles. Shells are not part of the OS; they are command interpreters. UNIX has several shells: Bourne, C, and Korn shell to name a few. If you do not like one shell, you can switch to another with a single command.

We will concentrate on the **concepts** of a user interface, **advantages**, and **disadvantages**; not on the specific commands or their syntax and usage of a particular interface. We will discuss the specific interfaces, the command line interface (CLI) for Windows and Unix, during the lecture on DOS and Unix as well as Labs 1 through 5.

PURPOSE OF THE USER INTERFACE

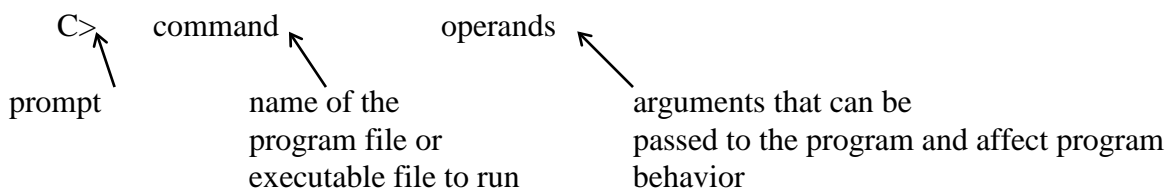
- (1) Make the facilities of the computer system accessible to the user in an efficient, convenient and friendly manner.
- (2) Provide user interface services to application programs that assure that different programs have user interfaces that operate in the same way - have **common look and feel**. (It reduces learning time.)

Windows 95/98/NT/2000/XP/Vista/7 - (MS Office 2007 suite has similar user interface to its earlier versions)

USER FUNCTIONS AND PROGRAM SERVICES

(1) Loading and Execution of Program Files

- (a) Using commands from the command line interface (CLI)



Ex. C>Program1	<E>	Windows
\$ Program1	or \$./Program1	Unix

(These must be executable files)

(b) Using the mouse from the graphical user interface (GUI), double-click on the icon which is a graphical representation of a program or a data file

(c) Use batch mode to run programs non-interactively

(2) File Commands

(Enable one to retrieve, store, and manipulate data files)

Factors important to the user:

(a) The ability to access data and programs by **name** – the user is not concerned with physical characteristics and representation of the file or its physical storage location

(Data can be in text form or binary form for images and sounds.)

(b) The ability to use commands that enable the user to **store, copy, move, manipulate, and retrieve files**

(c) The ability to issue commands that construct and **manipulate directories/folders** to organize files

Windows XP

Create new directory on the Desktop

- write click
- select New
- select Folder
- default folder name: "New Folder"
- rename the folder

(d) the ability to **redirect input and/or output** to different devices and files from their usual location

(e) ability to construct **pipes** and **filters**

Ex. Windows

Ex. Unix/Linux

type

cat

point 2b above

dir

ls

“

copy

cp

“

erase or del

rm

“

md

mkdir

point 2c above

rm

rmdir

C>*program1* < data1

\$ *program1* < data1

point 2d above

<code>type joe > joe1</code>	<code>cat joe > joe1</code>	“
<code>sort /r <random.dat>randsor.dat</code>	<code>sort -nr <random.dat>randsor.dat</code>	“
<code>dir >dirlist</code>	<code>ls -al > dirlist</code>	“
<code>dir more</code>	<code>ls -al more</code>	point 2e above

Output redirection	-	>
Input redirection	-	<
Piping	-	

The details are performed by the **file management system**.

(3) Disk and Other I/O Device Commands

- (a) **formatting** and **checking** disks
- (b) **copying** entire disks
- (c) **mounting** (adding) and **unmounting** (detaching) of devices to/from the system
- (d) **spooling** output

Windows: *format*, *chkdsk*, *diskcopy*

(4) Security and Data Integrity Protection

Files are **protected** from being read, deleted, copied to someone else, written to, or executed.

In Unix/Linux an owner of the file can control who has **access** to the file. There are several levels of security for each file: **read**, **write**, **execute**, and none (-); i.e., **r**, **w**, **x**, and **-**; and three groups of users: the owner, the owner's group, and everyone else. (See lecture notes on Unix/Linux and Labs 3-5).

-rwx-----	1	jmzura01 unixuser	7195	Mar 5 21:19	a.out
-rw-----	1	jmzura01 unixuser	151	Mar 9 17:18	.bash_history
-rwx-----	1	jmzura01 unixuser	33	Jan 20 14:15	.bash_logout
-rwx-----	1	jmzura01 unixuser	176	Jan 20 14:15	.bash_profile
-rwx-----	1	jmzura01 unixuser	194	Jan 20 14:15	.bashrc
-rwx-----	1	jmzura01 unixuser	720	Mar 5 21:22	designmenu
-rw-----	1	jmzura01 unixuser	12	Mar 4 17:00	joe
-rw-----	1	jmzura01 unixuser	0	Mar 4 17:04	joe1
-rw-----	1	jmzura01 unixuser	12	Mar 5 21:11	joe2
-rw-----	1	jmzura01 unixuser	12	Mar 9 16:48	joe3
drwx-----	4	jmzura01 unixuser	4096	Jan 20 14:15	.mozilla
-rw-----	1	jmzura01 unixuser	451	Mar 5 21:19	prog1.c
-rw-----	1	jmzura01 unixuser	1056	Mar 5 21:19	random.dat
-rw-----	1	jmzura01 unixuser	1057	Mar 5 21:20	randsor.dat
-rw-----	1	jmzura01 unixuser	450	Mar 5 21:35	savewho

The user can use *chmod* and *chgrp* commands to change permissions to the files.

On IBM/z system access is controlled by maintaining **Access Control Lists** (matrix).

Access rights define what access various **subjects** have to various **objects**.

Access rights:

- display (D)
- none (N)
- read (R)
- write (W)
- execute (E)

Subjects:

- users
- programs

Objects:

- files
- tapes
- disks
- data bases

Data access controls can be maintained by the table (matrix with the following elements) similar to this one.

		user name	user name	program name
	Subjects:	jmzura01	aachisz1	program1
Objects:				
lab1.pptx		N	R, W, E	E
employee.accdb		R, W, E	R	D, E



(filenames)

(5) **Inter-user Communication, Data and Program Disk Sharing Operations**

Most systems provide a means to pass or share data between programs and to communicate between users and programs

Editors, compilers, linkers, interpreters, data files, and OS system utilities (a sorting routine) can be shared by placing them in the common area (accessible to anyone) of disk storage.

E-mail, ftp, telnet, ssh, talk/chat (instant messaging) services.

Allow multiple users to work on the same document.

Provide piping operations to allow a program to communicate with one another.

The pipe command takes the output from one program and uses it as the input to another.

Redirection operations (< - input, > - output)

(6) **System Status Information** - available on most systems

(a) system administrators or system programmers

collect statistics on who uses the system resources (CPU time, main memory, disk storage space, I/O channels), when and how much in order to improve the performance of the system

(b) programmers and users have commands available to determine the amount of available disk space and memory, the number of users on the system and who they are, and the percentage of time that the CPU is busy

Ex. IBM z/system

Ex. Unix

Windows

QDISK, QSTORAGE

who, who am I,
quota

CHKDSK, DEFRAG

(7) **Program Services Known as Application Program Interface (API)**

Concern services, which the OS provides directly to programs, such as:

- storing, retrieving, and locating files
- allocating disk space
- performing and queuing I/O requests
- sharing and exchanging data
- distribute program processing among different machines on a network

are provided directly to the user's programs and are invisible to the user.

TYPES OF USER INTERFACE

(1) **Command Line Interface (CLI)** – Windows command prompt, Unix/Linux, batch system commands

Windows

C>command operand1 operand2 ... <Enter>

C>DIR/W - "/" is a switch


C>COPY C:\GAMES\SOLITAIR.EXE E:

"\" indicates a root
directory, "C:" - drive
designation

source destination

Positional parameters - derive the meaning from their position in the operands' field

Keyword parameters - derive the meaning from the keyword


C>MODE COM1 BAUD=2400 PARITY=N

(The MODE command allows one to change the parameters of the serial port such as COM1. BAUD determines the transmission rate and PARITY sets the error-checking mode. N stands for NONE.)

Ex. Windows

Ex. Unix/Linux

sort /r <random.dat>randсор.dat sort -nr <random.dat>randсор.dat

The **order** of **keyword parameters** in the command **can be changed**. The **order** of **positional parameters cannot be changed**.

Batch files, wild cards (Labs #1 through #5.)

Advantages: more flexible and powerful than other interfaces, consumes little memory, can combine commands, faster for experienced users, can use wild cards to apply a command to multiple files or directories

Disadvantages: must know the commands to use them, sometimes unfriendly (Unix: ls, cat), difficult to learn and use

(2) **Menu-driven Interfaces**

IBM AS-400, Earlier versions of Windows, UNIX on the HPCC at U of L.

Log in to Unix and run a script file name *designmenu* from the korn shell. Also, see lab #5.

./designmenu

You will see the following simple menu.

MAIN MENU

- 1) Print current working directory
- 2) List all files in current directory
- 3) Print today's date and time
- 4) Exit

Please enter your selection jmzura01:

1 <E>

Menu of alternatives is presented on the screen. The user can select an alternative:

- by moving a cursor with the mouse and clicking on the desired choice, or
- by typing in a number or a function key that corresponds to the choice.

It is convenient to inexperienced users. However, paging through multiple menus may be awkward. Shortcuts are available.

(3) **Graphical User Interface (GUI)**

Windows 3.11, Windows 95/98/NT/2000/XP/Vista/7, X Window, Mac OS X

The mouse-driven, icon-based Windows interface. Easy to learn and easy to use. Intuitive, effective. Amenable to multi-tasking.

Allows one to implement a multi-tasking system in which the user can control every task by placing each executing task in a separate window. The user can switch between tasks quickly, choosing one task at a time to be executed in the foreground, while others can run in the background.

Allows one to arrange the desktop windows according to your own preferences. Hard to implement, much more **demanding in its hardware and software requirements**, requires powerful video capability, requires a lot of memory to store pictures and programs. Software is complex, large, slow, and difficult to write. Has less flexibility and power than CLI. For example, using GUI it is more difficult to implement piping. Many experienced users prefer CLI.

SOFTWARE CONSIDERATIONS

CLI and Menu-driven Interfaces

Implemented in text mode display. Each menu option or each command in CLI is interpreted and directly connected to the OS module that will perform the requested service.

GUI

Moves of the mouse generate interrupts to the CPU. The position (X,Y coordinates) of the mouse needs to be computed each time the mouse is moved. Icons represent documents, devices or programs. Double-clicking on a document icon requires the software to identify an icon from its location on the display screen, determine the associated application program, load and execute the application, and load the document data. Windows can be redrawn, moved, copied, stored, resized, opened, closed.

GUI is fast and convenient to use when the computer and display are located together, such as in a PC.

Read about the X Window concepts.

COMMAND AND SCRIPTING LANGUAGES

1. Windows

Two of the Lab # 2 batch files.

```
:START
REM ... This is being typed by an endless loop.
GOTO START
```

```

ECHO OFF
IF %1==XYZ GOTO GOOD
ECHO BAD PASSWORD... ENDING
GOTO END
:GOOD
ECHO YOU'RE OK...STARTING
IF EXIST E:LOOP.BAT ECHO YES THERE IS A LOOP
PAUSE
REPEAT RED BLUE GREEN
:END

```

2. IBM System/z – MVS/JCL

Generic JCL

These control statements are interpreted and executed, one a time, to control the processing of a multistep job.

```

JOB parameters
EXEC PROG1 parameters
    DD define input file, parameters
    DD define output file, parameters
EXEC PROG2 parameters
    DD define input file, parameters
    DD define output file, parameters
.....
    (one DD statement for each input/output file)
END JOB

```

3. Unix – HPCC at U of L – script file from Lab 5 to implement a simple menu-driven interface.

```
#!/usr/bin/ksh
```

```
leave = no
```

```
while [$leave = no]
```

```
clear
```

```
do
```

```
cat <<++
```

MAIN MENU

- 1) Print current working directory
- 2) List all files in current directory
- 3) Print today's date and time
- 4) Exit

Please enter your selection \$LOGNAME:

```
++
```

```
read selection
```


case \$selection in

1)

pwd

echo "Press Enter to continue."

read junk

;;

2)

ls -al | pg

echo "Press Enter to continue."

read junk

;;

3)

date

echo "Press Enter to continue."

read junk

;;

4)

exit 0

;;

***)**

echo "Invalid choice. Try again."

echo "Press Enter to continue."

read *junk*

;;

esac

done

exit 0