History

We are now going to look at some history: this is useful as it will illustrate the many important parts of OSs and why they are necessary

History

We are now going to look at some history: this is useful as it will illustrate the many important parts of OSs and why they are necessary

At first, computers had no operating systems (1960s)

History

We are now going to look at some history: this is useful as it will illustrate the many important parts of OSs and why they are necessary

At first, computers had no operating systems (1960s)

- Every programmer had to write their programs for the particular machine they were using

History

We are now going to look at some history: this is useful as it will illustrate the many important parts of OSs and why they are necessary

At first, computers had no operating systems (1960s)

- Every programmer had to write their programs for the particular machine they were using
- So no portability

History

We are now going to look at some history: this is useful as it will illustrate the many important parts of OSs and why they are necessary

At first, computers had no operating systems (1960s)

- Every programmer had to write their programs for the particular machine they were using
- So no portability
- And lots of repeated code between programs (“write a character to the teletype”)

History

We are now going to look at some history: this is useful as it will illustrate the many important parts of OSs and why they are necessary

At first, computers had no operating systems (1960s)

- Every programmer had to write their programs for the particular machine they were using
- So no portability
- And lots of repeated code between programs (“write a character to the teletype”)

Remember: the more we make programmers do, the more likely they are going to make a mistake

History

Furthermore, programmers rarely even saw the computer

History

Furthermore, programmers rarely even saw the computer

- The program and the data (collectively called a *job*) would be prepared on paper tape or punched card

History

Furthermore, programmers rarely even saw the computer

- The program and the data (collectively called a *job*) would be prepared on paper tape or punched card
- Jobs would be given to operators who would the load and run them, then give the results back

History

Furthermore, programmers rarely even saw the computer

- The program and the data (collectively called a *job*) would be prepared on paper tape or punched card
- Jobs would be given to operators who would the load and run them, then give the results back
- Usually, there would be a bug and the programmer would have to fix the program and go round the loop again

History

Furthermore, programmers rarely even saw the computer

- The program and the data (collectively called a *job*) would be prepared on paper tape or punched card
- Jobs would be given to operators who would the load and run them, then give the results back
- Usually, there would be a bug and the programmer would have to fix the program and go round the loop again
- As computer time was limited, there was an issue of *scheduling* jobs, initially done by hand

History

Furthermore, programmers rarely even saw the computer

- The program and the data (collectively called a *job*) would be prepared on paper tape or punched card
- Jobs would be given to operators who would the load and run them, then give the results back
- Usually, there would be a bug and the programmer would have to fix the program and go round the loop again
- As computer time was limited, there was an issue of *scheduling* jobs, initially done by hand
- Turnaround on jobs could be days

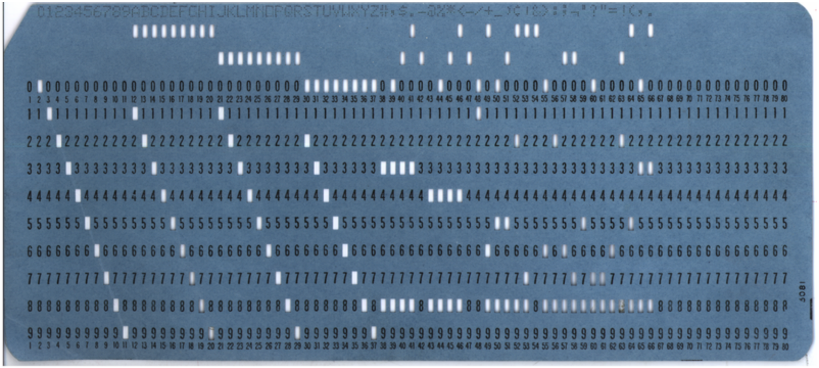
History

Furthermore, programmers rarely even saw the computer

- The program and the data (collectively called a *job*) would be prepared on paper tape or punched card
- Jobs would be given to operators who would the load and run them, then give the results back
- Usually, there would be a bug and the programmer would have to fix the program and go round the loop again
- As computer time was limited, there was an issue of *scheduling* jobs, initially done by hand
- Turnaround on jobs could be days

This concentrated the programmer's mind wonderfully!

History



From Wikipedia. Encodes a single 80 character line

History

From Wikipedia. 5 and 8 hole paper tapes

History

It was soon found there was a lot of repeated code between programs, so useful tools (programs and libraries of code) were developed to help manage repetitive tasks

History

It was soon found there was a lot of repeated code between programs, so useful tools (programs and libraries of code) were developed to help manage repetitive tasks

- collecting common functions in *system libraries* (sqrt, open file, etc.)

History

It was soon found there was a lot of repeated code between programs, so useful tools (programs and libraries of code) were developed to help manage repetitive tasks

- collecting common functions in *system libraries* (sqrt, open file, etc.)
- program management (loaders)

History

It was soon found there was a lot of repeated code between programs, so useful tools (programs and libraries of code) were developed to help manage repetitive tasks

- collecting common functions in *system libraries* (sqrt, open file, etc.)
- program management (loaders)
- debuggers

History

It was soon found there was a lot of repeated code between programs, so useful tools (programs and libraries of code) were developed to help manage repetitive tasks

- collecting common functions in *system libraries* (sqrt, open file, etc.)
- program management (loaders)
- debuggers
- Interfacing to hardware: I/O drivers (send file to printer, etc.)

History

It was soon found there was a lot of repeated code between programs, so useful tools (programs and libraries of code) were developed to help manage repetitive tasks

- collecting common functions in *system libraries* (sqrt, open file, etc.)
- program management (loaders)
- debuggers
- Interfacing to hardware: I/O drivers (send file to printer, etc.)

This made programming and program management easier, but there was still lots of human intervention needed

History

The issue was to keep the big and expensive computer as busy as possible, running programs all the time

History

The issue was to keep the big and expensive computer as busy as possible, running programs all the time

Idle time was a waste of money

History

The issue was to keep the big and expensive computer as busy as possible, running programs all the time

Idle time was a waste of money

So the operators would load many programs on to a fast medium, such as magnetic tape, and the computer would load and run them as fast as hardware allowed

History

The issue was to keep the big and expensive computer as busy as possible, running programs all the time

Idle time was a waste of money

So the operators would load many programs on to a fast medium, such as magnetic tape, and the computer would load and run them as fast as hardware allowed

This was called *spooling*, the first instance of addressing the disparity between human and computer speeds

History

Spooling would also be used on output: the output would be written to a mag tape, which could then later be attached to a printer

History

Spooling would also be used on output: the output would be written to a mag tape, which could then later be attached to a printer

Again, this was because printers are slower than computers

History

Soon it became clear this could be automated: have a little program, called a *monitor* (or *supervisor*), that loads and runs programs and puts the results somewhere sensible

History

Soon it became clear this could be automated: have a little program, called a *monitor* (or *supervisor*), that loads and runs programs and puts the results somewhere sensible

This would be directed by a *job control language*

History

A famous job control language from IBM was called JCL

History

A famous job control language from IBM was called JCL

Of course “JCL” means “Job Control Language”, but JCL was just one of a few job control languages

History

```
//IS198CPY JOB (IS198T30500),'COPY JOB',CLASS=L,MSGCLASS=X
//COPY01   EXEC PGM=IEBGENER
//SYSPRINT DD  SYSOUT=*
//SYSUT1   DD  DSN=OLDFILE,DISP=SHR
//SYSUT2   DD  DSN=NEWFILE,
//             DISP=(NEW,CATLG,DELETE),
//             SPACE=(CYL,(40,5),RLSE),
//             DCB=(LRECL=115,BLKSIZE=1150)
//SYSIN    DD  DUMMY
```

(From Wikipedia) Any guesses?

History

```
//IS198CPY JOB (IS198T30500),'COPY JOB',CLASS=L,MSGCLASS=X
//COPY01   EXEC PGM=IEBGENER
//SYSPRINT DD  SYSOUT=*
//SYSUT1   DD  DSN=OLDFILE,DISP=SHR
//SYSUT2   DD  DSN=NEWFILE,
//             DISP=(NEW,CATLG,DELETE),
//             SPACE=(CYL,(40,5),RLSE),
//             DCB=(LRECL=115,BLKSIZE=1150)
//SYSIN    DD  DUMMY
```

(From Wikipedia) Any guesses?

This copies OLDFILE to NEWFILE

History

```
//IS198CPY JOB (IS198T30500),'COPY JOB',CLASS=L,MSGCLASS=X
//COPY01   EXEC PGM=IEBGENER
//SYSPRINT DD  SYSOUT=*
//SYSUT1   DD  DSN=OLDFILE,DISP=SHR
//SYSUT2   DD  DSN=NEWFILE,
//             DISP=(NEW,CATLG,DELETE),
//             SPACE=(CYL,(40,5),RLSE),
//             DCB=(LRECL=115,BLKSIZE=1150)
//SYSIN    DD  DUMMY
```

(From Wikipedia) Any guesses?

This copies OLDFILE to NEWFILE
This would be set on 9 punched cards

History

A Fortran program, with data:

```
//CONVERT JOB USER=UGA001,MSGCLASS=6,NOTIFY=UGA001
//*MAIN CLASS=NITE,LINES=40,ORG=UGAIBM1.LOCAL
// EXEC FORTVCLG,REGION=2000K
//FORT.SYSIN DD *
       READ(5,10) CENT
    10 FORMAT(F6.2)
       FAHR=(CENT*9.0/5.0)+32.0
       WRITE(6,20) CENT,FAHR
    20 FORMAT(F6.2,' CENT = ',F6,2,'FAHR')
       STOP
       END
/*
//GO.SYSIN DD *
100.00
/*
//
```

History

JCL also allowed

History

JCL also allowed

- different *classes* of job: some people are allowed more time or memory space than others

History

JCL also allowed

- different *classes* of job: some people are allowed more time or memory space than others
- automatic *accounting*: who to charge for what. Charges would be made for CPU time, memory usage and anything else they could think of (another recurrent issue in CS)

History

JCL also allowed

- different *classes* of job: some people are allowed more time or memory space than others
- automatic *accounting*: who to charge for what. Charges would be made for CPU time, memory usage and anything else they could think of (another recurrent issue in CS)
- programmers could specify things like *when* they want the program to run, how much disk or memory it needs, etc. E.g., at certain times of day it might be cheaper to run a program

History

JCL also allowed

- different *classes* of job: some people are allowed more time or memory space than others
- automatic *accounting*: who to charge for what. Charges would be made for CPU time, memory usage and anything else they could think of (another recurrent issue in CS)
- programmers could specify things like *when* they want the program to run, how much disk or memory it needs, etc. E.g., at certain times of day it might be cheaper to run a program

If a job ran out of its allotted time or space it would be killed

History

JCL allowed several programs to be collected and loaded together in a single bunch

History

JCL allowed several programs to be collected and loaded together in a single bunch

This is called *batch processing*

History

JCL allowed several programs to be collected and loaded together in a single bunch

This is called *batch processing*

Running in batches is more efficient, as we spend more time running our programs and less time messing around in the overheads of loading and unloading

History

This might seem like ancient history, but these things are still happening

□

History

This might seem like ancient history, but these things are still happening

Modern large computers (like Bath's Balena cluster) are managed in just this way: and for the same reasons

□

History

This might seem like ancient history, but these things are still happening

Modern large computers (like Bath's Balena cluster) are managed in just this way: and for the same reasons

We still run jobs; charges are made for time and memory; and so on

□

History

This might seem like ancient history, but these things are still happening

Modern large computers (like Bath’s Balena cluster) are managed in just this way: and for the same reasons

We still run jobs; charges are made for time and memory; and so on

Turnaround is seconds or minutes rather than days, but the principle is the same

□

History

This might seem like ancient history, but these things are still happening

Modern large computers (like Bath's Balena cluster) are managed in just this way: and for the same reasons

We still run jobs; charges are made for time and memory; and so on

Turnaround is seconds or minutes rather than days, but the principle is the same

Exercise: look up *Portable Batch System*, PBS and compare with JCL

□