

Dartmouth Timesharing

1960 Dartmouth has an LGP-30 vacuum tube computer

Used for testing if F_{10} is prime (took 1 week)

Scoring ski meets

Searching for answers to some number theory questions

An ALGOL 60 compiler was written

Two pass + runtime

DOPE - Dartmouth oversimplified Programming Experiment

One freshman class used this program

Using NSF money Dartmouth purchases General

Electric 225 (upgraded to a 235) and Datatnet 30 computers and designs a timesharing system

Principles of the First Timesharing System

Designed for users unfamiliar with computers

Each user was independent of other users

Flexibility was sacrificed for simplicity and accomodating a large number of users

Largest file was 6000 characters

Very simple command set

All details of how the system hidden from user

User Model of the System

After logging on with a 6 digit user number the user was ready to type in a program and run it

Each user/terminal had a temporary program that could be edited

if the command "run" was typed then this program was run

There was a current name associated with this temporary file

The command "save" would save the current program under the current user number and name

The command "old" would retrieve a previously saved program to the temporary area

typing "stop" or "hello" at any time would stop everything and, if the command was "hello", log in a new user

All commands started with a letter

All lines starting wioth a digit were assumed to be part of the current program and added to it

From the user's point of view the lines of the program were always sorted

The last line with duplicate line numbers replaced previous similarly numbered lines

A line containing only a line number was deleted (with all similarly numbered lines)

The "run" command normally ran the basic system but other systems could be selected via the "system" command

Only the first three letters of a command were used

Initial User Command List

bootstrap*

bootstrapped the entire system

bye

log off of the system

catalog

print a list of saved files for this user

dial*

communicate with a specific user

dump*

dump DN30 memory

edit

call special edit package

goodbye

log off of the system

hello

log off previous user, reset tables, and prepare to logon a new user

keyboard

length

print the approximate length of the current program

list

list the current program

monitor*

eavesdrop on a specific user

new

create a new temporary file

notice*

put a notice message for new users

number*

change user numbers

octal*

patch DN30 memory

off*

turn off the timesharing system

old

retrieve a saved file

on*

turn on the timesharing system

rename

change the name of the current program

rff*

read a record from disk

run

If the program's lines have not been sorted - sort them

If the program has not been compiled - compile it

Start the program running

save

save the current file on the disk under the current name

scratch

set current program size to zero leaving everything else the same

status

report all status of this terminal and name of current file, user number etc.

stop

stop the running program

system

enter a system to run (basic, algol etc.)

s

stop

tape

prevent line feeds for inputing a paper tape

test

run a basic teach program - concatenate the current program with the corresponding teach program and start the result at line number 10000

tty

print status of tty
unsave
delete the saved program with the current program's name
users
print number of users currently using system
warn*
print a message on all teletypes
wrf*
write a record from disk

System Architecture

The GE 235 ran all compilers and user programs

All scheduling was done by the DN-30

The GE 235 just followed directions

- load this program
- load this system
- run this program
- edit this file
- etc.

The DN-30 could read and write the GE 235 memory and cause a program interrupt

Communication between machines was done by "mailboxes" in the 235 memory

If the DN-30 felt that the 235 was not responding properly

- It filled all of memory with BRU 26 instructions causing a branch to location 26
- It then wrote a bootstrap program in memory to restart the 235

GE 235 Memory Layout

There were two banks of 8K in memory

The longest disk read was 6K

The low 2K was reserved for the executive

The next 6K contained the "read only" part of the system or runtime

- If two consecutive user programs used the same system or runtime then this part was not changed

The next 6K contained the variable part of the user program

- The low words in this segment were the output area and were copied to the teletype after every swap

The top 2K of memory were also reserved for the executive

Datanet 30 Architecture

18-bit realtime communications processor

Q register counted down each machine cycle

- when it got to 0 the program was interrupted
- when it got to -32 the machine was bootstrapped from a paper tape reader

Teletypes were interfaced to "bit buffers"

Bits came at a rate of 110 per second

- Had to service bit buffers at least this fast
- Hardware SCN instruction would do all bit processing
- had to service the results 10 per second to send or receive characters

DN-30 program was divided into "real time" and "spare time"

- Real time serviced the teletypes
- Had to finish before next interrupt of the machine malfunctioned

Spare time did everything else
important tables in the machine were indexed by the "C" register that also selected peripherals

State for each user or teletype

- user number (2 words)
- problem name (2 words)
- disk address of region
- continue computing flag
- service flag
- input disk pointer
- output disk pointer
- special input flag
- input beginning of line pointer
- input disk buffer not ready
- input routine too slow
- input character pointer
- input word pointer
- input disk buffer address
- buffer service flag
- tape input flag
- save/old/unsave/catalog flag
- idle time
- header line needed
- line number (two words)
- line status
- error message table
- output buffer character pointer
- output delay counter
- output buffer not ready
- output buffer word pointer
- output after swap flag
- D-30 is outputting flag
- priority
- scheduling queue
- running time
- saved system name
- spaces after output
- standard disk addresses
- status
- swap length
- system
- test flag
- partial input buffer flag

Retrospective

Much of the software architecture was dictated by the hardware

Putting the control in the DN-30 acted as a memory protection

The DN-30 was probably the major reason for success

- Designed for communication and not computation

- Use of teletypes instead of computer terminals also more user friendly

Hardware constrained the flexibility of the system to a system that was implementable

- No grand designs could be considered

John Kemeny's continual insistence for beginning user friendliness stopped a lot of computer scientist hacks

DTSS Second Generation

Better (or at least bigger) hardware

More experience

Opportunity to fail a few times

Ken Lochner continually pushed the "right" abstractions

Probably a major reason for the success of the second system

Inventor of "communication files" that later became UNIX pipes