Multics: Product, Project, and Development Process

- **Introduction / history**
- **The development process**
- **Architecture and other important technical topics**
- **Summary**

A lecture given by John W Gintell to the Greater Boston ACM / IEEE Computer Society meeting on November 16, 1989

Introduction / History

- Capsule summary
- Attributes of system
- Current status / how used
- Evolutionary summary
- Overall project goals – established in 1965

Capsule Summary

- <u>Multi</u>plexed <u>I</u>nformation and <u>C</u>omputing <u>S</u>ervice
- Conception in 1964/65
- Joint MIT / General Electric / Bell Labs project
- Operational in 1968/69
- Gradual project transfer to Honeywell (GE) personnel
- Fully supported product in 1973
- Growing customer base / many kinds of use through 1987
- Major evolution from 1969 through 1987
- Wind–down through 1990's …

Attributes of system

- Symmetrical multiprocessor and I/O
- Segmented, paged virtual memory / storage system
- Shared data and inter-process communications
- Security at B2 level
- Stack, reentrant procedures, inter-language calling
- Dynamic linking and environment shaping
- Callable system interfaces
- Flexible terminal I/O – dumb and slightly smart

- Consistent, rich command language
- Extremely wide range of commands
- Rich spectrum of services
- Most conventional languages
- Relational database manager

- Enterprise system orientation
- Flexible / distributed administration and operation

Current status / how used

- General purpose time-sharing
- Information and communications utility
- Software factory
- Production TP with and without RDB
- Special purpose applications

Evolutionary summary

- 13 major releases
- 115 major internal versions
- Replacement of almost everything several times
- Simplification and generalization
- Adaption of concepts from rest of industry
- 4M lines of code, uncounted pages of documentation

Overall project goals – established in 1965

- Remote terminal access as normal mode of usage
- Continuous operation
- Wide range of growth capacity
- Reliable file system
- Control of access to allow selective sharing
- Hierarchically structured information and administration
- Serve small and large users efficiently
- Support different user environments
- Flexibility and generality of system for evolution

The development process

- Project philosophy
- Use of electronic communications
- The Multics Change Review Board
- Code review
- Quality assurance methodology

Project philosophy

- Architectural foundation
- Emphasis on specification and high-level design
- Documented in publication quality English
- Maintain consistency and coherency
- Principal of least surprise
- Plan / develop for evolution and maintenance
- Code itself should be suitable for publication
- Development process consistent with system
- Use system to build itself
- Continuous integration of components
- Encourage the use and construction of tools
- Extensive use of peer review
- Promote group ownership of system

Use of electronic communications

- Group geographically split (Cambridge, Phoenix, Calgary)
- Communication among all project members with email
- On-line design review with electronic meetings
- All documentation and source code available on-line
- Trouble reporting and error tracking
- Distribution of critical fixes
- Connection to outside world in the Internet

The Multics Change Review Board

- Reviews all proposed changes, big and small
- Review is on user-visible interface and effects, not design
- Evolution of process
  - Several strong individuals
  - A hand picked board
  - Cooperating boards in two locations
  - Anyone can join board
  - Electronic implementation
  - Executive board to resolve disputes

Code review

- Started as a grass roots effort
- Informal standards written
- Increasing responsibility of auditor
- Apply to all software and documentation
- New software and minor changes
- Process formalized for B2 definition

Quality assurance methodology

- No formal system test group or final qual group

- High quality code and documentation
- Configuration management applied to entire system
- Developer owns the problem and the solution
- Peer pressure to conform to quality standards
- Architecture of system itself
- Bootstrapping nature of system

Architecture and other important technical topics

- Virtual memory / process architecture
- Multics process
- Security
- Measurement / metering / tuning
- PL/I relationship
- Data hiding
- Error handling
- Parameterization
- Simplification
- Storage system evolution
- User interface evolution
- Distributed administration
- Dynamic reconfiguration

Virtual memory / process architecture

- All storage objects in segmented virtual memory
- Segments are paged, transparently to users
- Segments have pathnames
- Standard interface is: get a pointer, refer to data directly
- All data is directly addressed (down to the bit)
- Access control on each data reference
- Operating system runs in same address space as user
- Rings used to protect OS
- Device independent I/O system permits segment/file reference with different programming interface (not normally used in commands)
- Problems
  - Segment size limitation (1MB) led to multi-segment files
  - File reference to data outside of system
  - Asynchronous writing makes DB recovery difficult
- File manager for relational database manager to have been included inside system later

Multics Process

- Large address space
- Permanent for life of login (can get new process)
- Environment / static storage shared between commands
- Dynamic linking results preserved
- Powerful, but expensive to create
- Single execution point
- Restricted multi-tasking available

Security

- Principal of least privilege used extensively
- Discretionary access control: ACLS
- Rings for system structuring
- Mandatory access control: Access Isolation Mechanism
- Most functionality for B2 since 1975
- Mailboxes and message segments with per-message access
- Generalization of extended access
- Audit logs

Measurement / metering / tuning

- Metering policy
  - If it can be counted, put in a meter
  - Build commands to display results
- Parameterize algorithms with commands to change
- Segment / page usage counting
- Paging/ disk IO / interrupts / system faults /scheduler metering
- Call tracing
- Statement profiling
- Usable by users for debugging and performance analysis

PL/I relationship

- System programming language == principal user language
- PL/I call interface universally used
- OS primitives form a PL/I machine
- Reliance on PL/I character string and pointer
- Use of area mechanism
- Special interface required for other languages

Data hiding

- Programmable interfaces for system data
- Subsystem interfaces with appropriate argument lists
- Pointers to PL/I structures used to pass information
- Structure version number placed in structures
- Callee checks version number for compatibility handling
- Allows stage evolution of interfaces

Error handling

- Error codes or signaling
- Originally: signaling with an error stack
- Actually: status code passed to callers
- Error state not kept within the system
- Standardized system status codes
- Programs refer to code tables symbolically
- Using dynamic linking to get to message table
- Status codes actually pointer
- Yields user supplied error tables
- Signals for specific errors / situations
- Command argument error condition
- Command question condition and the answer command

Parameterization

- Customization without system modification
- System administrator defined tables for
  - Numbers, names, relationship of objects
  - Metering and tuning parameters
  - Terminal attributes
- File transfer and printing services
- Per-project/user supplyable overrides where appropriate

Simplification

- Approach to abysmal system performance
- Static binding, segment combining
- Recode into subset of PL/I for optimal compiling
- 3 rings to 2 for standard operations
- Use of fixed size items
  - Compile-time sizes
  - Pages and segment sizes
  - Directory branch and ACL sizes
  - Fixed character string sizes
  - Fixed binary instead of bit
- Watch “word” boundaries
- Don’t recode into assembly language or destroy modularity
- Later improvements removed shortfalls
  - Better compiler
  - More registers in hardware
  - Rings in hardware
  - Byte and bit addressing in hardware

Storage system evolution

- Original: entire storage system on one logical volume
- Separation into logical volumes with multiple physical volumes
- VTOC on physical volume for efficiency and recovery
- Demountable volumes
- Directories on separate logical volume
- Volume and hierarchy backup/retrieval/reload systems
- Various forms of fault tolerance added
- Salvagers and scavengers

User interface evolution

- Commands with control arguments
- Long and short names
- Standardized control arguments
- Extensive command file language: exec_com
- Abbrev, active functions, iteration, and do command
- Help files for all commands
- Primitive subroutines available to ease command writing
- Provide subroutine interfaces for principal functions
- Yields better error handling, modularity and efficiency
- Subsystem_utils for commands with family of requests
- Window and video system for dumb terminals
- Menu interface to a few commands

Distributed administration

- System tables controlled separately
- I/o devices and communications configuration
- User registration
- Storage management
- Printing and backup services
- Project setup and resource grouping
- Project administrators control projects
- User and delegatable privileges
- Resource usage
- Restricted subsystems
- Priority scheduler
- Workclasses and load control groups

Dynamic reconfiguration

- Add / remove hardware components while system is running
- CPUs, IO controllers, memories
- Front End Processors
- Communication lines and I/O devices
- Divide 1 system into 2 and put it back together again
- Partition subset to run online T&D
- Add / remove software components while system is running
- All but the prelinked at boot-time OS itself

Summary

- Effect on Bull (who absorbed the Honeywell computer business)
- Effect on world of computing
- Missing or uncompleted
- An evolutionary project

Effect on Bull

- Multics is current world-wide communications vehicle
- 3 of top 10 US Bull customers heavy Multics users
- Many ideas / concepts in Bull products started in Multics
  - 2 of 3 major proprietary lines have segmenting/ring architecture
  - 3rd has common common hardware except for addressing architecture
  - Many features replicated in other systems
  - Command interface of mini-computer line is like Multics
  - Proprietary RDB of large system ideas based upon Multics
- Multics was Software Factory for two product lines
- Multics development process adopted in portions of company

Effect on world of computing

- Many current proprietary OS use concepts and interfaces
- UNIX™ origins
- UNIX™ futures
- ~1,000 members of Multics development community exported

Missing or uncompleted

- Mechanism for modern user interface
- Tasking / light-weight processes
- SNA and OSI networking
- Large native files
- Distributed file system
- A good C environment with UNIX™ compatibility

An evolutionary project

- Original objectives met
- Most components replaced several times
- Product consistency maintained across many changes
- Still in active use as enterprise system at a number of customers
- Strong evidence that significant evolution of a product possible
- Project success, but product failure
- It was way ahead of its time
- Didn't mesh well with company strategy
- Almost all users and developers really liked it: this is rare!
- Many new ideas tired – a number of firsts

- Multics should be viewed as a prototype of many ideas in computing