usp lab.

Unicage Development Method Technical Overview

Universal Shell Programming Laboratory, Ltd.

January 2013

Universal Shell Programming Laboratory

What is Unicage?

usp lab.

It's a software development method using Linux, text files and shell scripting to build information systems cheaply, quickly and flexibly.

Low Cost

Low Cost / Easy to Program

- Uses inexpensive PC-based servers and OS (Linux)
- Data is plain text, program is shell script, no middleware required
- No forklift upgrades (data and software remain same when hardware and OS change)

Fast

Fast Development Time / Fast Processing

- Programs are very short
- One engineer can design, develop and operate
- We've stripped out unnecessary features to maximize hardware performance

Flexible

Flexible

- Program is simple and easy to customize
- Software functions are not dependent on each other so can be changed easily
- Data necessary for the application is created from organized, structured data

Our Method Follows the Unix way of Thinking

usp lab.

The UNIX Philosophy has been Unchanged for 40 Years

1. Small is Beautiful
2. One program (command) should only do one thing
3. Prototyping should be as fast as possible
4. Portability takes precedence over efficiency
5. Data is stored as plain text
6. Commands are used as “levers” (can be combined & reused)
7. Applications are written in shell script
8. All programs are designed as filters (pipes)

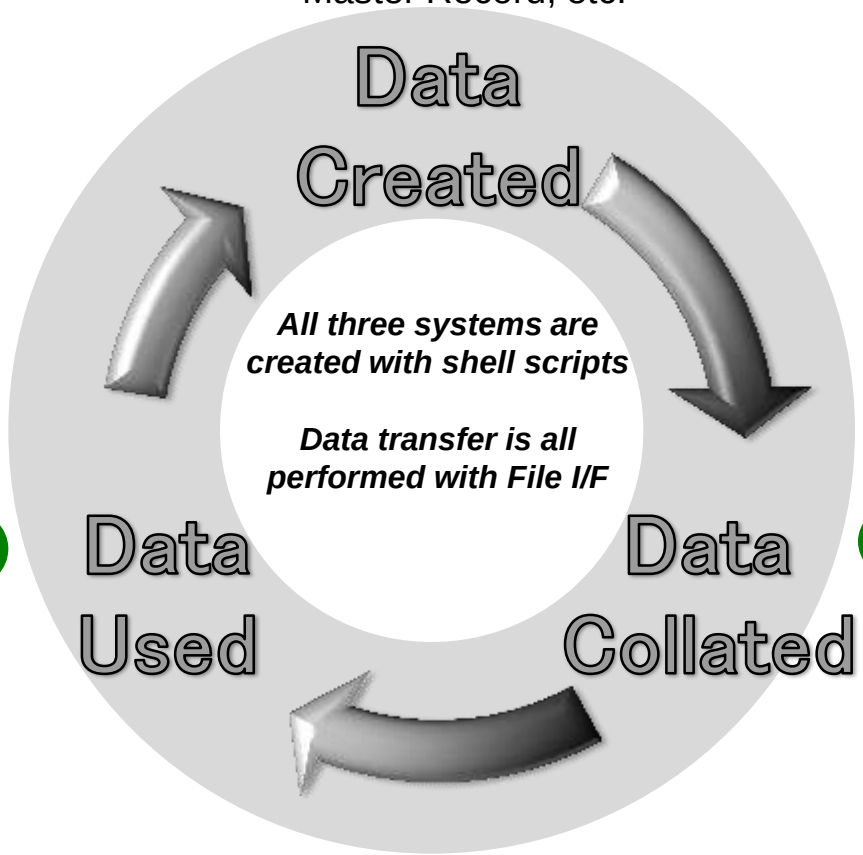
Source: “The UNIX Way of Thinking” by Mike Gancarz

Unicage Software Architecture

usp lab.

Input Script

POS
Order Data
Master Record, etc.



Output Script

Screen
Report, etc.

Update/Collate Script

Data merged
5W1H Collation
5 Layer Data Management, etc.

One-Dimensional Interface Prevents "Compartmentalization" of System Development

Universal Shell Programming Laboratory

Unicage Philosophy (Fully Distributed)

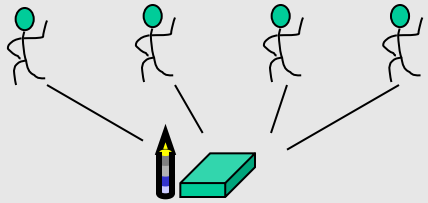
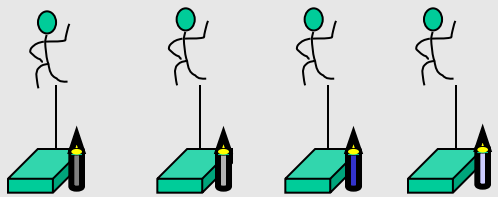
usp lab.

Separate by Business, Separate by Organization
Data, Program and Hardware are all Separate

① Separation reduces waste, improves processing speed.
② With roles separated, maintenance becomes easy.

To **Separate** is to **Understand**

Non-Shared data is Held Separate
Shared data is Fully Shared (original data is distributed)

Traditional "Sharing" (centralized)	Full Sharing (distributed)
	
One change of spec affects everyone	One change of spec doesn't affect others
High Load / Program is Complex	Low Load / Simple Program

Universal Shell Programming Laboratory

Improve the Effectiveness of the System

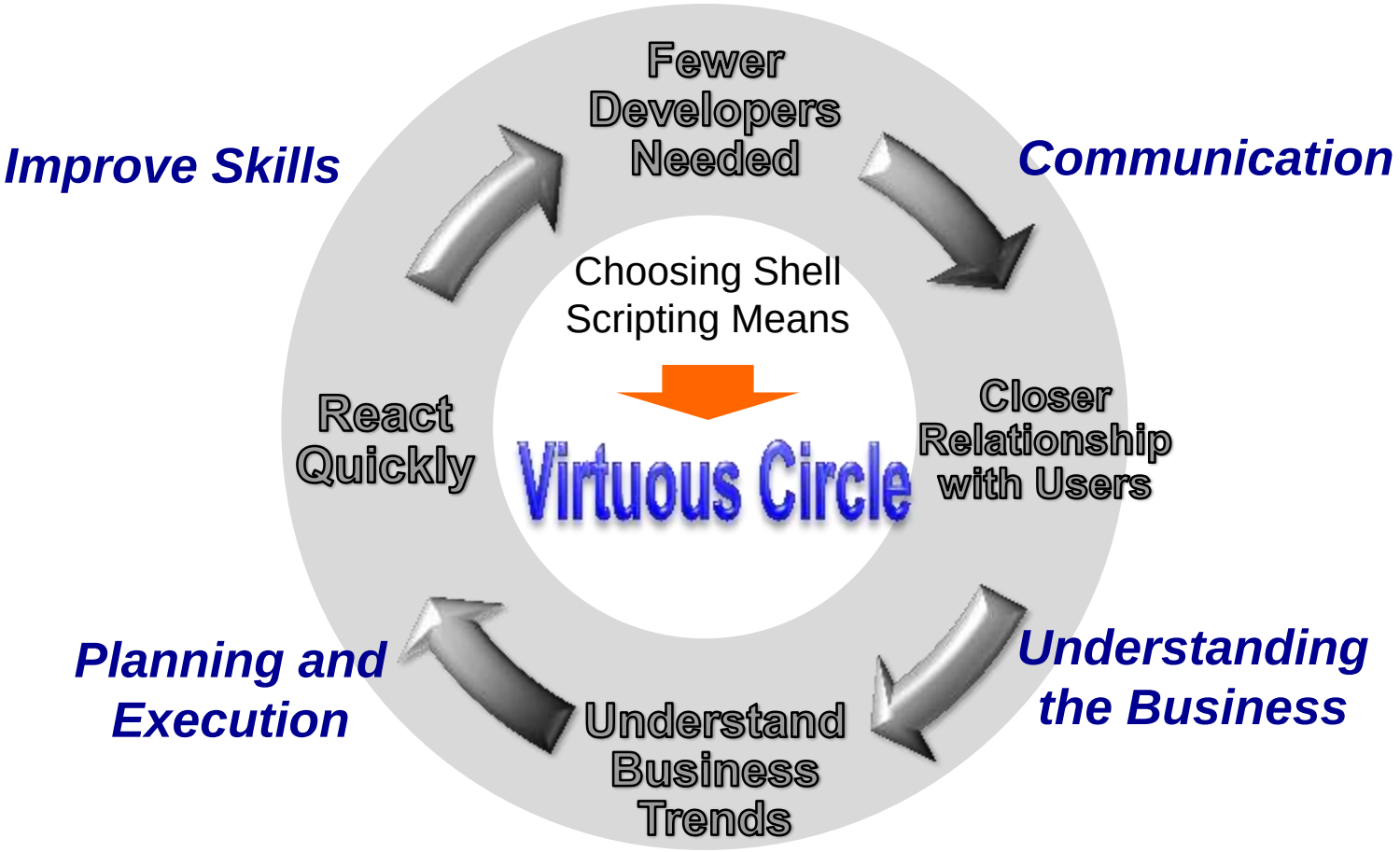
usp lab.

Developer's Perspective

Program
Requirements

User's Perspective

Effectiveness
Business Need



Universal Shell Programming Laboratory

Areas for using Unicage Development Method

usp lab.

◎ Best Cases

→ Fast processing of large data sets (Summing, Searching, Reporting)

→ Transaction processing such as order management

→ Support for internationalization (UTF-8 Text Data)

△ Borderline Cases (Where Internal Development is not Optimum)

→ Legal, accounting, etc use cases based on social rules (Financial Accounting, Salary Calculations, etc.)

→ Areas where software packages are already very functional

× Not Recommended (Extremely Challenging)

→ Device Drivers

→ Complex User Interfaces

- However, perfect for a data interface using HTTP

→ iPad, Andoroid, RIA or .NET web applications

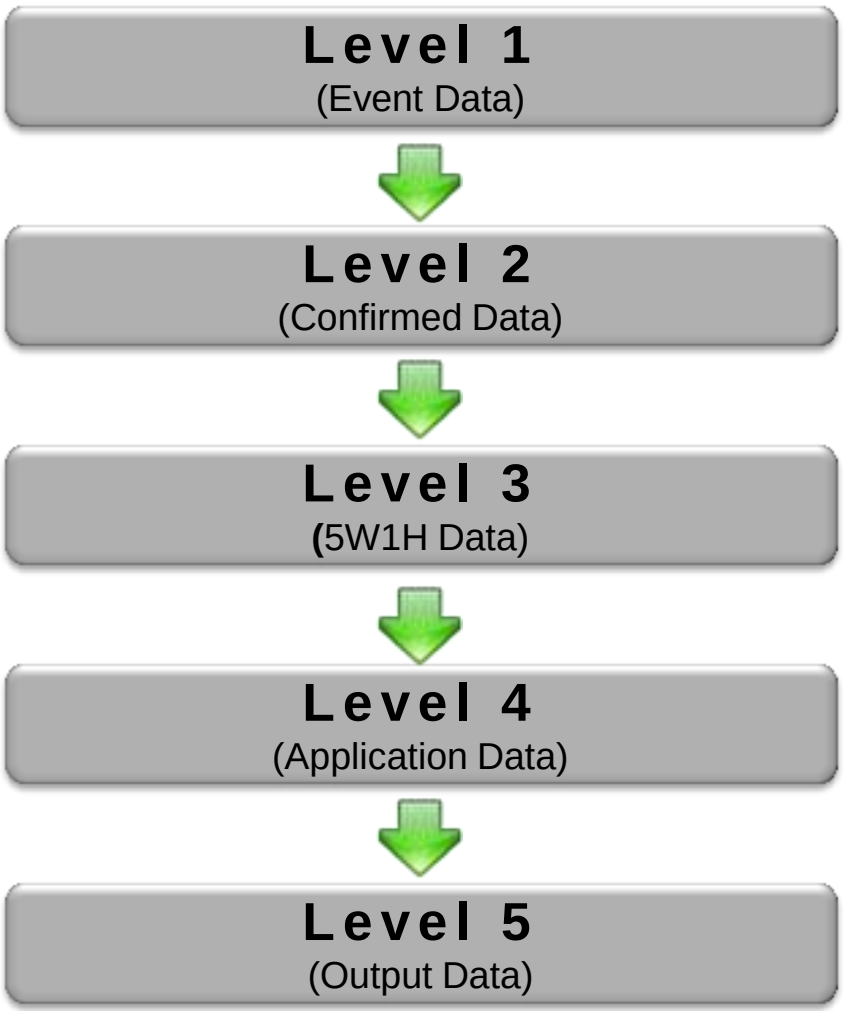
× Drag and Drop GUIs
3D calculations for polygons

Universal Shell Programming Laboratory

Unique Data Management Technique

Layered Data Management

usp lab.



Raw Data (Unsynchronized)
Data created as the information is produced
(Must not be lost; replicated and backed-up robustly)

Detail Data (Synchronized)
Raw data that has been formally imported into the system at a particular point in time

Level 2 data is organized and summarized in 5W1H units
(Often called “Ultra-organization.” Does not depend on the application.

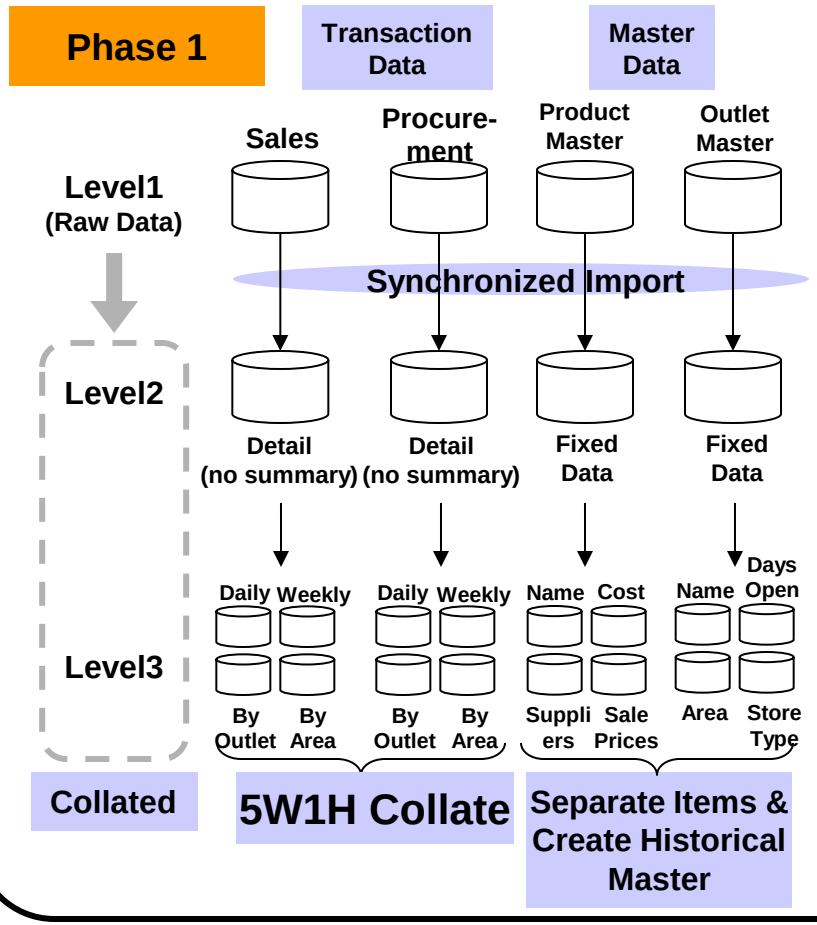
Data formatted to be easily processed by the application

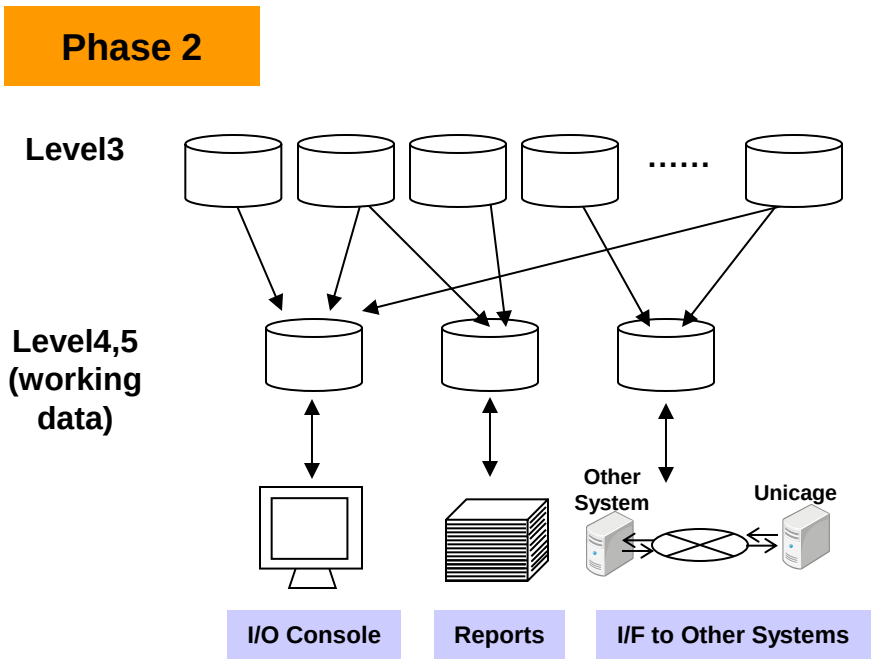
Report Image Data (Excel, PDF) or Log

Universal Shell Programming Laboratory

Development Process

→ **Phase 1 (Building the Data Base) and Phase 2 (Building the Application) must be done in this order.**





First create a system for importing the raw data and collating it (Phase 1)

Build the application system (Phase 2)

Unicage Development Method 2

Application Development Process

70% Important for Business

30% Important for Business

Design Framework for Business

=

Can be Decided Early

Initial Development

Obtain Real Data
Develop, Test, Release

Can be designed only with the original data and output format design (Design Workload is Light)

Design Useful Features

Based on Working Application so Easy to Design

Second Pass (Finishing)

Develop, Test, Release

Short Program
＋
No Dependencies
↓
Feature Extension is Easy

Short Timeframe

Few Man-Hours

Short Timeframe

Few Man-Hours

Short Program ＋ **User Focus** ＋ **2-Steps** = **Low Cost Fast Response**

Fast Development

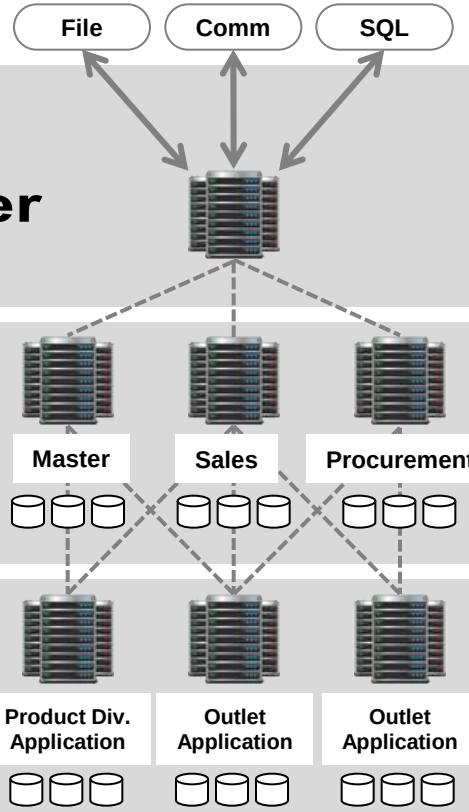
Efficient Communication

Accurate Spec Design

usp lab.

Universal Shell Programming Laboratory

Server Architecture



① I/F Server

- Servers to send and receive data to/from external systems (per source/protocol)
- Data is not stored; simply passed through

② Storage Servers

- Servers that collate from L1 to L3 and store data for each business. (By data type)

Servers are scaled out with master/slave architecture

③ Service Servers

- Servers to support applications for each business unit. (Per Unit)
- Has Web, data and application software functions

Servers are scaled out with master/slave architecture

- Each server group has policies for the data and shell scripts stored on it, therefore management is unified regardless of the system type or size.
- By ensuring hardware policies are unified, hardware selection and operations is kept flexible.

Unicage Commands

usp lab.

All programs are filters ⇒ The clearest filter is a command

Only 100 Common Commands

cp	Copy	Linux Original Commands
find	File search	
sort	Sort	
awk	Perform operations on items	
⋮		
join0	Data matching	usp Unicage Commands
sm2	Sumup	
waku	Add a border	
ulock	Lock control	
⋮		
mdate	Date management	Custom Commands

- Independent command set honed to single functions
- Simple to use (manual-less)
- Commands written in C, Python, Java, perl, awk, etc.
- Makes it simple to perform complicated business processing by combining simple commands.

Build a System with Shell Scripts (Batch Processing)

Combine Unicage (uspTukubai) commands and perform batch processing

```
#!/bin/bash
 join0 key=1 MASTER URE       |  Join data
 self 2 3 4 5                 |  Select field
 hsort key=1/2                |  Sort
 sm2 1 2 3 4                  |  Sum up
 sm4 1 1 2 2 3 4              |  Intermediate total
 self 1 2 4 3                 |  Select Field
 sm5 1 3 4 4                  |  Final total
 map num=1                    |  Transpose
 sed 's/A/Sales/g'            |  Text search/replace
 sed 's/B/Profit/g'           |     "
 keta 4 6@NF-1                |  Align rows
 comma 3/NF                   |  Add commas
 cat header -                 |  Attach header
 tocsv > result                  Output to CSV
 exit 0
```

→ Execute the shell script to process the data

Build a System with Shell Scripts (Transaction Processing)

Perform searches and updates from a web browser.

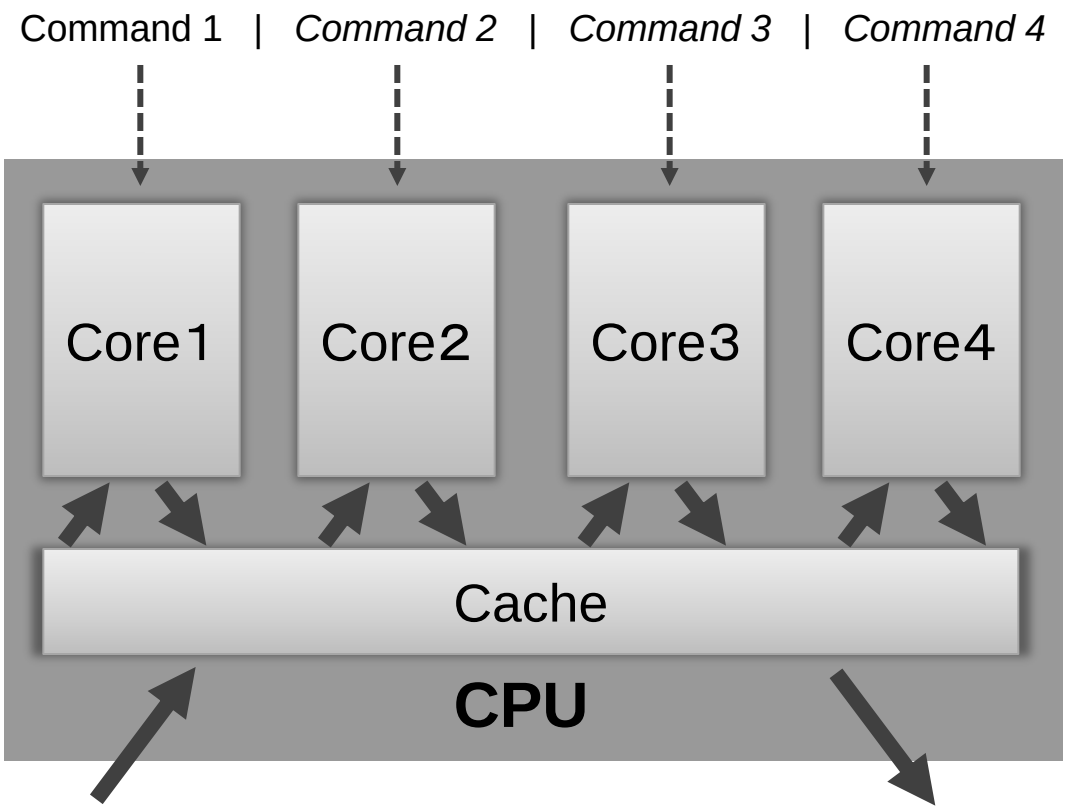
```
#!/bin/bash
dd bs=$CONTENT_LENGTH | cgi-name > name          # Get information from web server
case "$(nameread MODE name)" in                  # Branch based on processing mode
SEARCH)                                          # [Search]
    if ulock -r MST.LK; then                       # Shared Lock
        nameread KEY name     |                    # Get search key
        join0 key=1 - MST     |                    # Search for master data
        mojihame -lLABEL html                      # Export to HTML
    fi ;;
UPDATE)                                          # [Update]
    if ulock -w MST.LK; then                       # Exclusive lock
        nameread -el "KEY|VAL" name > TRN.123      # Get key and value
        upl key=1 MST TRN.123 > MST.123            # Create update master
        ln -s MST.123 MST                          # Allow access with same name
        cat next_html                              # Output to next screen
    fi ;;
esac
exit 0
```

→ For updates, the actual file is not updated, a newly-created file is referenced.

Use Multicore CPU Efficiently with Pipes

usp lab.

Shell Script Pipe Processing



- Distributed to cores by process
- Data is processed quickly in memory
- If you use named pipes you can program branches explicitly

Uses the kernel scheduler functionality as intended

Software Features (5)

How to Implement Specific Processes

Exclusive Lock

- Atomic processing commands using lock file/semaphore (Ex. flock, ulock commands)

Order Control

- Recursive server (Command execution agent) (Ex. server, ncat, rfifo commands)
- Loop Shell

Commit

- Make decision based on return value from OS system call (File System Reliability is equivalent to MySQL or PostgreSQL)

Rollback

- Data history is preserved so you can restore to any previous point (Temporal Database)

Parallel Processing

- Use “para” and “clust” command families

Access Spikes

- Install load balancing servers
  - Order Control
  - Resource Control by the OS
  - Scale Out (Add Servers)

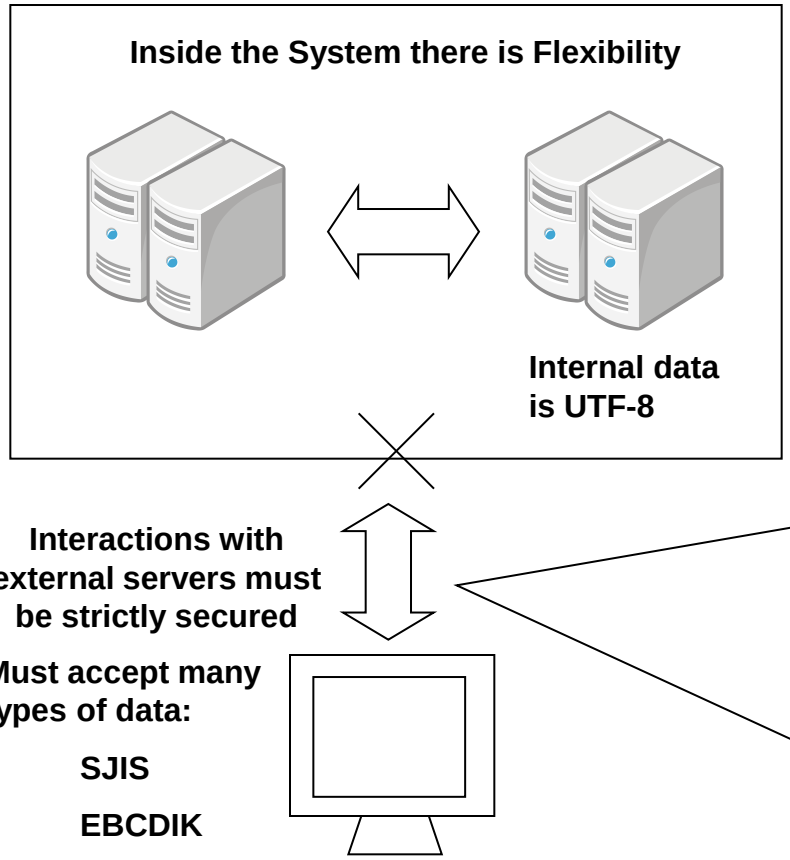
Universal Shell Programming Laboratory

How to Write Shell Scripts

Shell Scripts are extremely flexible so we must pay close attention to proper style when using Unicage

- Script header style
- Comment style
- Variable and file naming rules
- Rules for naming temporary files
- One command per line
- Transfer data using files (not environment variables)
- Include processing is forbidden
- File layout style
- Execution log style
- Rules for naming files
- Output execution start and end times
- Generate a semaphore file
- Keep it short
- Separate script and data in complex IF statements
- Delete garbage files
- Don’t create versions (but make backups)
- Multi-level calls prohibited
- Overwrite the copyright
- Understand the size of the processing file

Security Paradigm



[Example]

- **Use PAM for passwords**
- **Change passwords often (passwd --stdin)**
- **Restrict interactive logins**
- **Encrypt data and communications (MD5,GPG,SSL,etc.)**
- **Server Security (tcpwrapper/iptables/xinetd/chkconfig)**
  - **Snort, exec-shield**
  - **Use SELinux**
  - **Prevent command insertion from Web applications、**
  - **Prevent buffer overflows**
  - **Use firewalls**

❖ **When interacting with external systems, convert the character set and data layout.**

Operations and Monitoring

→Software requirements for operations and monitoring utilize OS functions and are written in the shell script.

- Execution Log
  - #!/bin/bash -xv
  - exec 2> LOGFILE
- System Log
  - “logger” command (syslog)
- Process Complete Alert
  - scp MESSAGE $remote_host:$dir
- Redundancy and Backup
  - H/W RAID, rsync, Load balancer
- Timed Execution
  - cron, at
- Order of Operations Control
  - Semaphore file
- Job Control
  - Background processing + wait
- Error Detection
  - $PIPESTATUS, “on err” function
- Periodic Reboot
  - reboot, fsck

Documentation

① Very Little Documentation is Required for Development

- **Configuration of data and programs is fixed, so only basic documentation is necessary.**
- **Required documents are as follows:**
  - Application I/O API specifications
  - Dimensions of source data

② Documentation for Understanding the System is Practical

- **“System Purpose”, “Business Flow”, “Manuals” are needed.**
- **Most information needed to understand the system can be obtained by looking at the system operation itself (examples below)**
  - Data configuration and relationships
  - Application configuration and relationships
  - Batch schedule
  - Detailed specifications (written in the shell scripts)

How to Make Unicage Effective for You

The following points must be clear:

1. You must be able to interface to the source data from the start of the project
2. You must understand the meaning and logic of all data items
3. You must assign members to the project who understand the business operations in detail
4. If performing internal development, make sure to train at least two dedicated engineers.
5. Develop as quickly as possible then quickly release regular updates