Linux firmware for iRMC controller on Fujitsu Primergy servers

Vladimir Shakhov
R&D Senior Development Engineer

Fujitsu Technology Solutions, R&D department

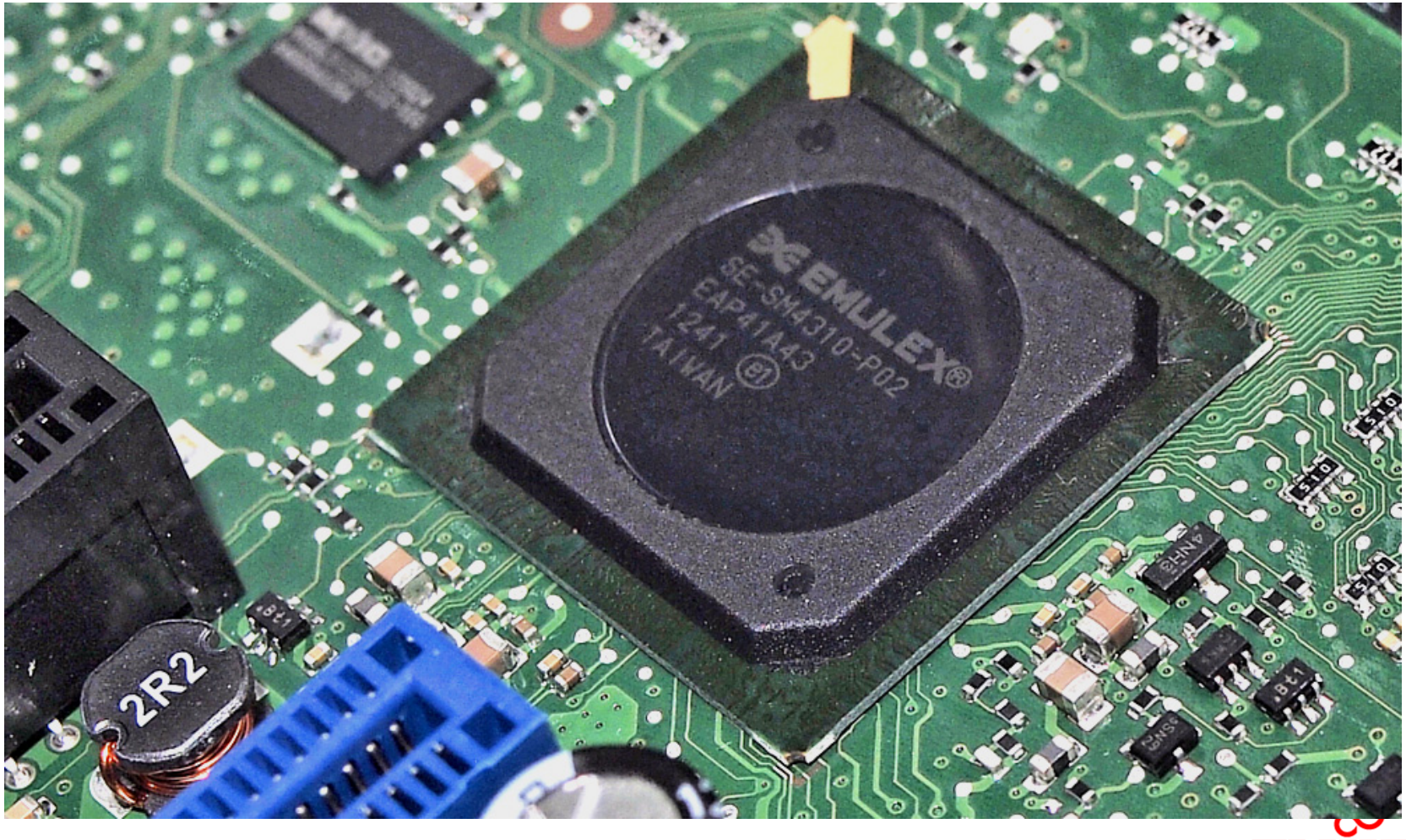
Table of content

FUJITSU

Fujitsu Primergy Servers

Lineage of x86-based servers:
Blade (BX), Rack (RX), Tower (TX) and Cloud (CX).

iRMC S4 in the wild



FUJITSU

iRMC - integrated Remote Management Controller

ARM-based SoC

Emulex Pilot3 iBMC ASIC

Integrated BMC

Super I/O

Graphics controller

KVMS: Remote Keyboard, Video, Mouse and Storage

CPU: 32-bit 400MHz ARM9 processor with MMU.

FUJITSU

iRMC - integrated Remote Management Controller

ARM-based SoC

Emulex Pilot3 iBMC ASIC

Integrated BMC

Super I/O

Graphics controller

KVMS: Remote Keyboard, Video, Mouse and Storage

CPU: 32-bit 400MHz ARM9 processor with MMU.

Control x86 hardware

Work independent if x86 host on or off.

FUJITSU

iRMC - integrated Remote Management Controller

ARM-based SoC

Emulex Pilot3 iBMC ASIC

Integrated BMC

Super I/O

Graphics controller

KVMS: Remote Keyboard, Video, Mouse and Storage

CPU: 32-bit 400MHz ARM9 processor with MMU.

Control x86 hardware

Work independent if x86 host on or off.

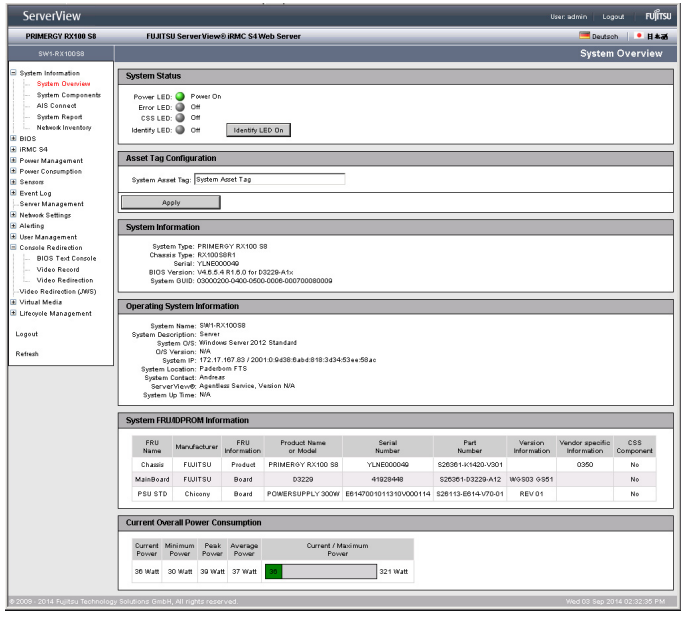
Own Operation System

Very typical Embedded Linux.

FUJITSU

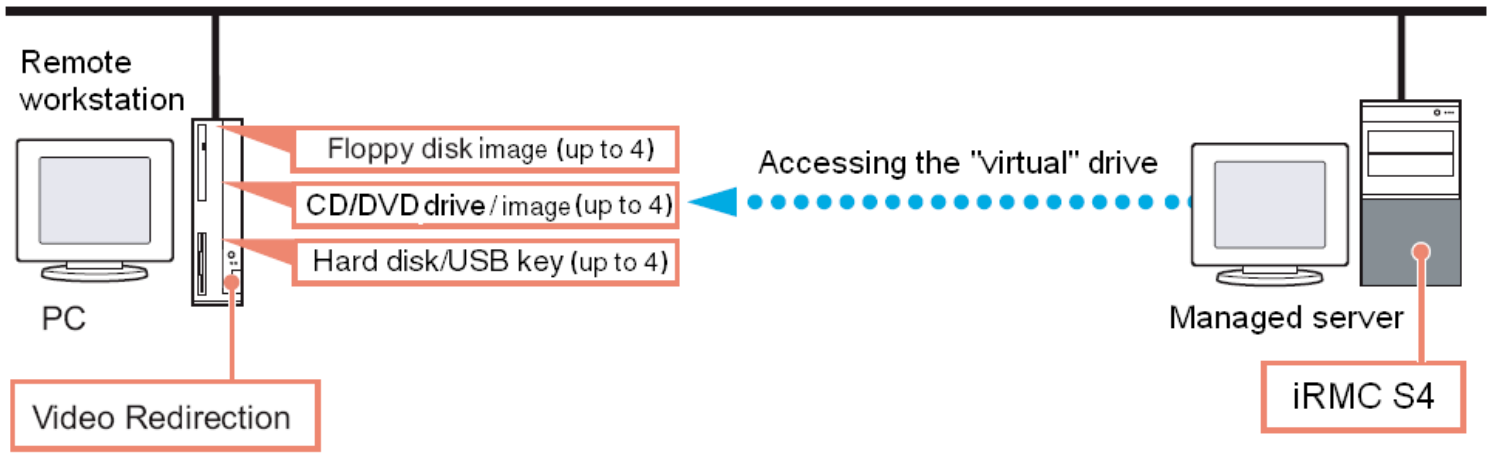
iRMC basic features

- Web access (own web-server)
- Security (SSL and SSH included)
- ServerView suite Integration
- Power management
- SNMPv1/v2c/v3 support
- Text console redirection
- “Headless” system operation
- CLP - command line interface



iRMC advanced features

- Advanced Video Redirection (AVR)
- Virtual Media
- Embedded Lifecycle Management (eLCM)



FUJITSU

Intelligent Platform Management Interface

IPMI - standardized, abstract, message-based interface between BMC and intelligent hardware for platform management. Key component of system.

Open standards

Intelligent Platform Management Interface

IPMI - standardized, abstract, message-based interface between BMC and intelligent hardware for platform management. Key component of system.

Web: HTTP, HTML, JavaScript

Web-based control interface.

Open standards

Intelligent Platform Management Interface

IPMI - standardized, abstract, message-based interface between BMC and intelligent hardware for platform management. Key component of system.

Web: HTTP, HTML, JavaScript

Web-based control interface.

SNMP ver 1/2x/3

Popular protocol for network management.

FUJITSU

Open standards

Intelligent Platform Management Interface

IPMI - standardized, abstract, message-based interface between BMC and intelligent hardware for platform management. Key component of system.

Web: HTTP, HTML, JavaScript

Web-based control interface.

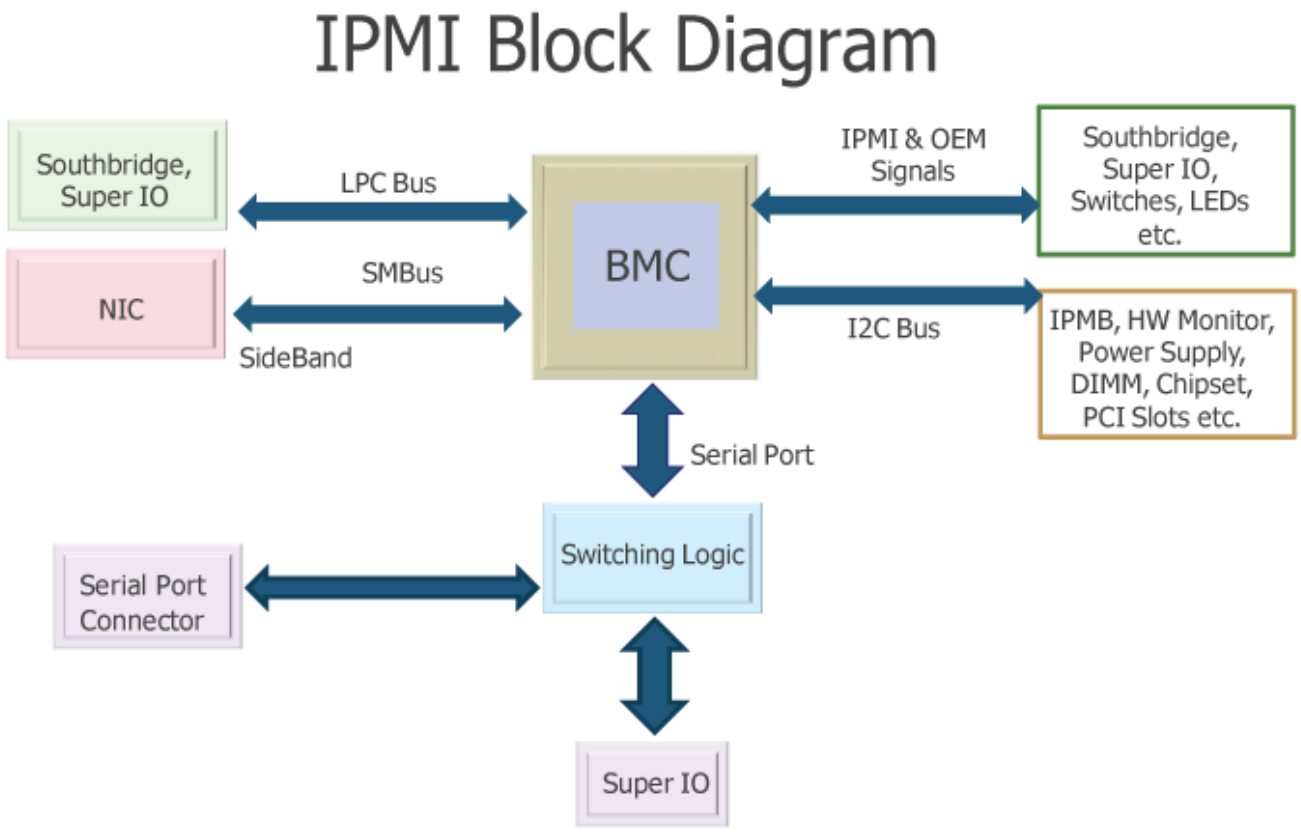
SNMP ver 1/2x/3

Popular protocol for network management.

Security: SSH and SSL

FUJITSU

IPMI - key interface of a system



FUJITSU

Demo 1

Web interface: AVR, VirtualMedia, remote boot

Scenario 1: AVR, show boot settings

AVR: show Windows, Start LCM Custom Image, AVR: Show Linux

FUJITSU

Demo 1

Web interface: AVR, VirtualMedia, remote boot

Scenario 1: AVR, show boot settings

AVR: show Windows, Start LCM Custom Image, AVR: Show Linux

Scenario 2: IPMI - via ipmitool

$ ipmitool -U admin -P admin -H 192.168.1.1 -I lanplus [command line]

command line variants:

- chassis status
- lan print
- user list
- sensor

FUJITSU

Pro

- Advanced Real-Time Operation System
- Small footprint
- Fast performance

FUJITSU

iRMC S1 - S2/S3 OS

Pro

- Advanced Real-Time Operation System
- Small footprint
- Fast performance

Contra

- Lack of available developers
- Lack of 3rd party ready components
- High cost of support
- Long features time-to-market
- Environment compatible only with themselves

FUJITSU

Why Linux

FUJITSU

Why Linux

Cost of development and support

- More developers available
- Huge amount of 3rd party ready components
- Faster development
- HW platform fast enough to run it

FUJITSU

Backward compatibility

- Same interfaces (UI, protocols)
- Binary firmware upgrades

FUJITSU

Backward compatibility

- Same interfaces (UI, protocols)
- Binary firmware upgrades

Code re-use

- OS API are different
- OS layout completely different
- HW-related stuff to rewrite from scratch

FUJITSU

Demo 2: OpenSSH + RemoteManager

Interface ~~bug to bug~~ byte to byte identical to ThreadX.

```
**************************************************
*    Welcome to PRIMERGY Remote Manager          *
*    Firmware Revision 98.10a (1.00)             *
*    SDR 3.16  ID 0401 TX1320M1                  *
*    Firmware built  Nov  5 2015 16:35:12 CET    *
**************************************************

System Type  : PRIMERGY TX1320 M1
System ID    : YLXLXXXX36
System Name  : SUT-PW
System OS    : Windows Server 2016 Technical Preview 3 Standard
System Status: OK (Identify LED is OFF)
Power Status : Off
Asset Tag    : System Asset Tag

    Main Menu

(1) System Information...
(2) Power Management...
(3) Enclosure Information...
(4) Service Processor...

(c) Change password
(*) Console Redirection (EMS/SAC)
(s) Start a Command Line shell...
(l) Console Logging

Enter selection or (0) to quit:
```

FUJITSU

iRMC Firmware components

Free and Open Source Software

- Linux Kernel
- U-Boot bootloader
- Busybox
- GNU Glibc
- Net-SNMP
- OpenSSH

FUJITSU

iRMC Firmware components

Free and Open Source Software

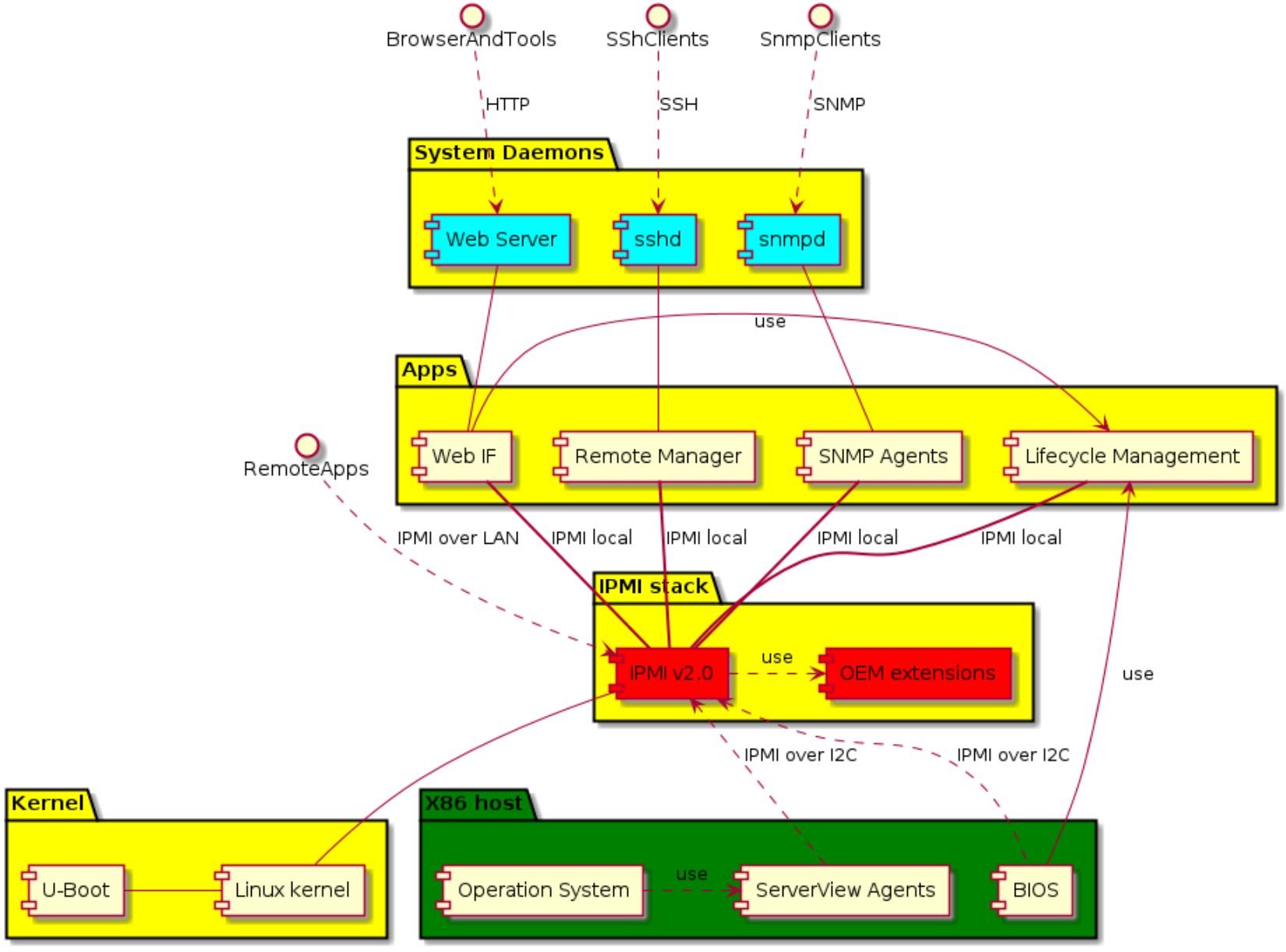
- Linux Kernel
- U-Boot bootloader
- Busybox
- GNU Glibc
- Net-SNMP
- OpenSSH

Closed source

- IPMI stack
- WebServer
- SNMP agents

FUJITSU

iRMC firmware internals



LXC containers + X2go for developers

The same environment for all to build and debug.
Read-only root filesystem on container.
Debian GNU/Linux based.

Custom package system

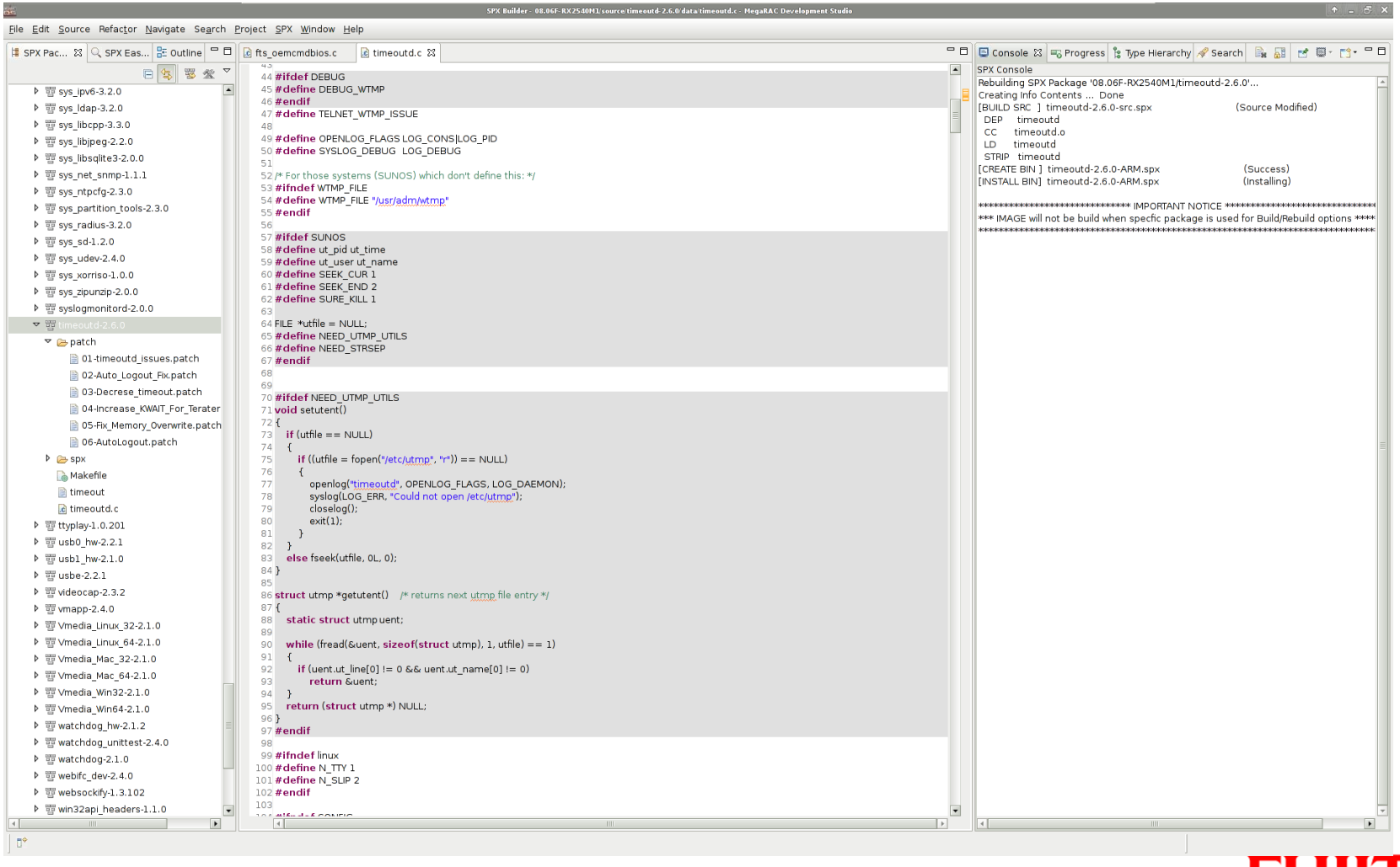
Used only in development and build process.
Not used for updates.
Package format similar to DEB, but not the same.

Eclipse-based IDE

Rich IDE + version control + packaging system integration

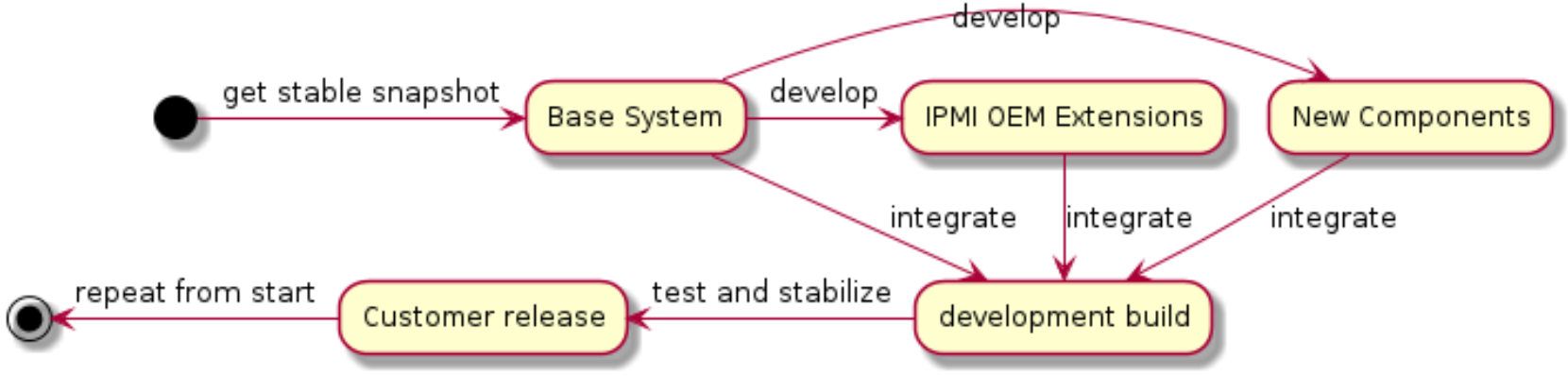
FUJITSU

IDE


SPX Builder - 08.06F-RX2540M1/source/timeoutd-2.6.0/data/timeoutd.c - MegaRAC Development Studio
File Edit Source Refactor Navigate Search Project SPX Window Help
SPX Pac... SPX Eas... Outline
sys_ipv6-3.2.0
sys_ldap-3.2.0
sys_libcpp-3.3.0
sys_libjpeg-2.2.0
sys_libsqlite3-2.0.0
sys_net_snmp-1.1.1
sys_ntpcfg-2.3.0
sys_partition_tools-2.3.0
sys_radius-3.2.0
sys_sd-1.2.0
sys_udev-2.4.0
sys_xorriso-1.0.0
sys_zipunzip-2.0.0
syslogmonitord-2.0.0
timeoutd-2.6.0
patch
01-timeoutd_issues.patch
02-Auto_Logout_Fix.patch
03-Decrese_timeout.patch
04-Increase_KWAIT_For_Terater
05-Fix_Memory_Overwrite.patch
06-AutoLogout.patch
spx
Makefile
timeout
timeoutd.c
ttyplay-1.0.201
usb0_hw-2.2.1
usb1_hw-2.1.0
usbe-2.2.1
videocap-2.3.2
vmapp-2.4.0
Vmedia_Linux_32-2.1.0
Vmedia_Linux_64-2.1.0
Vmedia_Mac_32-2.1.0
Vmedia_Mac_64-2.1.0
Vmedia_Win32-2.1.0
Vmedia_Win64-2.1.0
watchdog_hw-2.1.2
watchdog_unittest-2.4.0
watchdog-2.1.0
webifc_dev-2.4.0
websockify-1.3.102
win32api_headers-1.1.0
fts_oemcmdbios.c
timeoutd.c
#ifdef DEBUG
#define DEBUG_WTMP
#endif
#define TELNET_WTMP_ISSUE
#define OPENLOG_FLAGS LOG_CONS|LOG_PID
#define SYSLOG_DEBUG LOG_DEBUG
/* For those systems (SUNOS) which don't define this: */
#ifndef WTMP_FILE
#define WTMP_FILE "/usr/adm/wtmp"
#endif
#ifdef SUNOS
#define ut_pid ut_time
#define ut_user ut_name
#define SEEK_CUR 1
#define SEEK_END 2
#define SURE_KILL 1
FILE *utfile = NULL;
#define NEED_UTMP_UTILS
#define NEED_STRSEP
#endif
#ifdef NEED_UTMP_UTILS
void setutent()
{
if (utfile == NULL)
{
if ((utfile = fopen("/etc/utmp", "r")) == NULL)
{
openlog("timeoutd", OPENLOG_FLAGS, LOG_DAEMON);
syslog(LOG_ERR, "Could not open /etc/utmp");
closelog();
exit(1);
}
}
else fseek(utfile, 0L, 0);
}
struct utmp *getutent() /* returns next utmp file entry */
{
static struct utmp uent;
while (fread(&uent, sizeof(struct utmp), 1, utfile) == 1)
{
if (uent.ut_line[0] != 0 && uent.ut_name[0] != 0)
return &uent;
}
return (struct utmp *) NULL;
}
#endif
#ifndef linux
#define N_TTY 1
#define N_SLIP 2
#endif
Console Progress Type Hierarchy Search
SPX Console
Rebuilding SPX Package '08.06F-RX2540M1/timeoutd-2.6.0'...
Creating Info Contents ... Done
[BUILD SRC] timeoutd-2.6.0-src.spx (Source Modified)
DEP timeoutd
CC timeoutd.o
LD timeoutd
STRIP timeoutd
[CREATE BIN] timeoutd-2.6.0-ARM.spx (Success)
[INSTALL BIN] timeoutd-2.6.0-ARM.spx (Installing)
********** IMPORTANT NOTICE **********
*** IMAGE will not be build when specfic package is used for Build/Rebuild options ****
****

FUJITSU

Development cycle



Very typical Embedded Linux development cycle (simplified view):

1. Get base system snapshot and freeze it
2. Develop new components and IPMI OEM extensions
3. Bug fix and stabilization
4. Test it hard
5. Release firmware to customers
6. Repeat once again from step (1).

FUJITSU

FOSS legal questions

- Following the FOSS licenses
- Special policy for FOSS components using
- Consolidation of components legal status
- Rare upstream communication [1]
- FOSS component sources - by demand from support

FUJITSU

[1] no significant changes in upstream

Demo 3

Development login via SSH.
Show typical Embedded Linux system.

FUJITSU

Questions? Remarks?

- Fujitsu Primergy servers: http://www.fujitsu.com/fts/products/computing/servers/primergy/
- iRMC S4 manual: http://manuals.ts.fujitsu.com/file/11470/irmc-s4-ug-en.pdf
- Emulex Pilot 3 iBMC specs: http://www.emulex.com/products/controllers/management-controllers/pilot-baseboard-management-controller/specifications
- ThreadX RTOS: http://rtos.com/products/threadx
- Fujitsu Technology Solutions: http://www.fujitsu.com/fts

contact: Vladimir.Shakhov at ts.fujitsu.com

FUJITSU