

# TOPPERSカーネル拡張トレースログの 標準形式トレースログへの変換ルール

名古屋大学大学院情報科学研究科  
附属組込み研究センター 受託研究員

# 目次

---

- ❖ 先週の作業内容
- ❖ 標準形式変換ルールとは？
- ❖ ファイル構成
- ❖ 標準形式変換ルールの作成法
- ❖ 出力させたいリソースを決める。
- ❖ /resourceHeader/asp.reshに、リソースタイプの定義をする。
- ❖ /convertRules/asp.cnvに、標準形式変換ルールを書く。
- ❖ 置換マクロ
- ❖ tlv.tfに各リソースのtemplateを記述をする。
- ❖ tlv.tfの文法/実例
- ❖ TLVで、変換された標準形式ログを確認する。
- ❖ 今週の作業予定

# 先週の作業内容

---

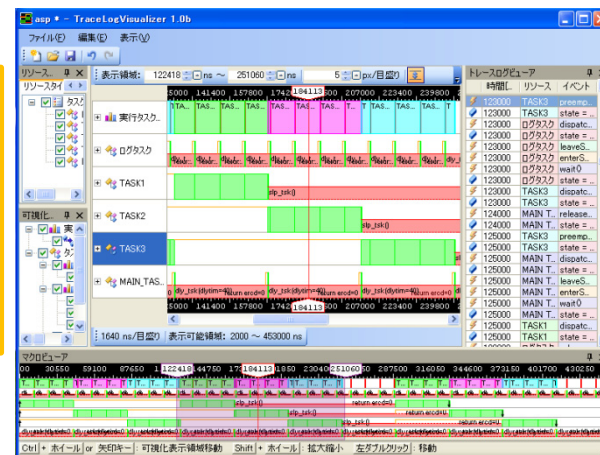
- ❖ 拡張トレースログの標準形式変換ルール作成
  - セマフォ、イベントフラグ、データキュー、  
優先度データキューのtemplate記述(tlv.tf)、  
変換ルール(asp.cnt)、リソース系(asp.res)作成。

# 標準形式変換ルールとは？

❖ OSが出力した様々のトレースログを、TLVが可視化のために、認識できるような形式にする規則。

```
1 [2000] dispatch to task 1.
2 [2000] enter to ena_int intno=2.
3 [2000] leave to ena_int ercd=0.
4 [2000] enter to dly_tsk dlytim=10.
5 [2000] task 1 becomes WAITING.
6 [2000] dispatch to task 5.
7 [2000] enter to act_tsk tskid=2.
8 [2000] task 2 becomes RUNNABLE.
9 [2000] leave to act_tsk ercd=0.
10 [2000] enter to act_tsk tskid=3.
11 [2000] task 3 becomes RUNNABLE.
12 [2000] leave to act_tsk ercd=0.
13 [2000] enter to act_tsk tskid=4.
14 [2000] task 4 becomes RUNNABLE.
15 [2000] leave to act_tsk ercd=0.
16 [2000] enter to dly_tsk dlytim=40.
```

今回は、標準形式  
変換について説明



標準形式変  
換

```
2 |LOGTASK.dispatch()",
3 |LOGTASK.state=RUNNING",
4 |LOGTASK.enterSVC(ena_int,intno=2)",
5 |LOGTASK.leaveSVC(ena_int,ercd=0)",
6 |LOGTASK.enterSVC(dly_tsk,dlytim=10)",
7 |LOGTASK.wait()",
8 |LOGTASK.state=WAITING",
9 |MAIN_TASK.dispatch()",
10 |MAIN_TASK.state=RUNNING",
11 |MAIN_TASK.enterSVC(act_tsk,tskid=2)",
12 |TASK1.activate()",
13 |TASK1.state=RUNNABLE",
14 |MAIN_TASK.leaveSVC(act_tsk,ercd=0)",
15 |MAIN_TASK.enterSVC(act_tsk,tskid=3)",
16 |TASK2.activate()",
```

可視化変換ルール

# ファイル構成

---

## ❖ tlv\_package\_1.0beta\_2009\_01\_13の内容

convertRules > **asp.cnv**, fmp.cnv

resourceHeaders > **asp.res**, fmp.res

visualizeRules > asp\_rules.viz, asp\_shapes.viz,  
fmp\_rules.viz, fmp\_shapes.viz,  
toppers\_rules.viz, toppers\_shapes.viz

❖ asp/kernel/**kernel.tf**

❖ asp/arch/logtrace/**tlv.tf**

❖ asp/my\_obj/semaphore/**kernel.res**

**\* 今回、見て頂きたいファイル。**

# ASP標準形式変換ルールの作成法

---

1. 出力させたいリソースを決める。
2. /resourceHeader/asp.reshに、リソースタイプの定義をする。
3. (リソースファイルを生成するTFファイルを作成。)
4. /convertRules/asp.cnvに、標準形式変換ルールを書く。
5. tlv.tfに、各リソースのtemplateを記述をする。
6. TLVで、変換された標準形式ログを確認する。

# 1.出力させたいリソースを決める。

```
switch (type) {
case TFN_EXT_SIG_SEM_WUP:
    info[0] = (intptr_t)trace->loginfo[1];
    info[1] = (intptr_t)trace->loginfo[2];
    tracemsg = "reference to ext_sig_sem_wup semid=%d. tskid=%d.";
    break;
case TFN_EXT_SIG_SEM_CNT:
    info[0] = (intptr_t)trace->loginfo[1];
    info[1] = (intptr_t)trace->loginfo[2];
    tracemsg = "reference to ext_sig_sem_cnt semid=%d. semcnt=%d.";
    break;
case TFN_EXT_ISIG_SEM_WUP:
    info[0] = (intptr_t)trace->loginfo[1];
    info[1] = (intptr_t)trace->loginfo[2];
    tracemsg = "reference to ext_isig_sem_wup semid=%d. tskid=%d.";
    break;
case TFN_EXT_ISIG_SEM_CNT:
    info[0] = (intptr_t)trace->loginfo[1];
    info[1] = (intptr_t)trace->loginfo[2];
    tracemsg = "reference to ext_isig_sem_cnt semid=%d. semcnt=%d.";
    break;
case TFN_EXT_WAI_SEM_CNT:
    info[0] = (intptr_t)trace->loginfo[1];
    info[1] = (intptr_t)trace->loginfo[2];
    tracemsg = "reference to ext_wai_sem_cnt semid=%d. semcnt=%d.";
    break;
case TFN_EXT_POL_SEM_CNT:
    info[0] = (intptr_t)trace->loginfo[1];
    info[1] = (intptr_t)trace->loginfo[2];
    tracemsg = "reference to ext_pol_sem_cnt semid=%d. semcnt=%d.";
    break;
case TFN_EXT_TWAI_SEM_CNT:
    info[0] = (intptr_t)trace->loginfo[1];
    info[1] = (intptr_t)trace->loginfo[2];
    tracemsg = "reference to ext_twai_sem_cnt semid=%d. semcnt=%d.";
    break;
case TFN_EXT_INI_SEM_CNT:
    info[0] = (intptr_t)trace->loginfo[1];
    info[1] = (intptr_t)trace->loginfo[2];
    tracemsg = "reference to ext_ini_sem_cnt semid=%d. semcnt=%d.";
    break;
}
```

```
[3000] dispatch to task 3.
[3000] enter to sig_sem semid=3.
[3000] reference to ext_sig_sem_cnt semid=3. semcnt=1.
[3000] leave from sig_sem ercd=0.
[3000] enter to act_tsk tskid=1.
[3000] task 1 becomes RUNNABLE.
[3000] leave to act_tsk ercd=0.
[3000] enter to act_tsk tskid=2.
[3000] task 2 becomes RUNNABLE.
[3000] leave to act_tsk ercd=0.
[3000] task 3 becomes WAITING.
[3000] dispatch to task 1.
[3000] enter to twai_sem semid=1.
[3000] leave from twai_sem ercd=0.
[3000] enter to ini_sem semid=1.
[3000] reference to ext_ini_sem_cnt semid=1. semcnt=1.
[3000] leave from ini_sem ercd=0.
[3000] enter to ext_tsk.
[3000] task 1 becomes DORMANT.
```

## 2./resourceHeader/asp.reshtに、リソースタイプの定義をする。

### ❖リソースタイプの定義

```
"リソースタイプ名": {  
  "DisplayName": 表示名  
  "Attributes": 属性  
  "Behaviors": 振舞い  
}
```

❖ 今回は、'セマフォ'のリソースを新しく追加。

```
"Semaphore": {  
  "DisplayName": "セマフォ",  
  "Attributes": {  
    "id": {  
      "VariableType": "Number",  
      "DisplayName": "ID",  
      "AllocationType": "Static",  
      "CanGrouping": false  
    },  
    "semcnt": {  
      "VariableType": "Number",  
      "DisplayName": "カウント",  
      "AllocationType": "Dynamic",  
      "CanGrouping": false  
    }  
  }  
  "Behaviors": { /*まだない。*/ }  
}
```



### 3.セマフォの標準形式変換ルールを作成

❖ [time] reference to ext\_xxxx semid=x. semcnt=x. xxx...



正規表現に直す。



❖ “¥[(?<time>¥d+)¥] reference to ext\_[^ ]+ semid=(?<id>¥d+). semcnt=(?<semcnt>¥d+) ..\*”



トレースログがこの正規表現に一致するとき

❖ “\$EXIST{Semaphore(id==\${id})}” の条件を満たすか判定。







条件を満たすとき

❖ “ [\${time}]\$RES\_NAME{Semaphore(id==\${id})}.semcnt=\${semcnt} ”を標準形式ログとして出力する。

# 置換マクロ

---

- ❖ `$EXIST{リソース}`       該当するリソースが存在したら、`true`を、存在しなかったら`false`をリターン。
- ❖ `$ATTR{属性}`       該当する属性に置き換えられる。
- ❖ `$RES_COLOR{リソース}`       `xx.res`で定義している、各リソースの色に置き換えられる。
- ❖ `$RES_NAME{リソース}`       リソース名に置き換えられる。

## 4.tlv.tfに各リソースのtemplateを記述をする。

---

- ❖今回は'Task'リソース以外に'Semaphore'のリソースを新しく追加するので'tlv.tf'ファイルにtemplateを記述が必要。
- ❖次のページからtlv.tfの文法を簡単に説明する。

# tlv.tfの文法

asp/arch/logtrace/tlv.tf

❖ **\$NL\$** ← 改行の出力

❖ **\$JOINEACH** semid **SEM.ID\_LIST**  
",¥n"\$ ← TABの出力

❖ **\$TAB\$**\$TAB\$"\$semid\$":{\$NL\$

❖ \$TAB\$\$TAB\$\$TAB\$"Type"  
:"Semaphore", \$NL\$

❖ \$TAB\$\$TAB\$\$TAB\$"Attributes"  
:\$NL\$

❖ \$TAB\$\$TAB\$\$TAB\${\$NL\$

❖ \$TAB\$\$TAB\$\$TAB\$\$TAB\$"id"  
:\$+semid\$, \$NL\$ ← VALUEの値

❖ \$TAB\$\$TAB\$\$TAB\$\$TAB\$"semcnt"  
"

:\$+SEM.ISEMCNT[semid]\$ \$NL\$

❖ \$TAB\$\$TAB\$\$TAB\$}\$NL\$ ← VALUEの文字列

❖ \$TAB\$\$TAB\$}

❖ **\$END\$**

\$で始まり\$で終わる:マクロ命令  
\$空白 で始まる :コメント  
何もなし :そのまま出力

すべてのオブジェクトにあるパラメータ  
**オブジェクト名.ID\_LIST**  
:割り当てられたID

繰り返し構文

**\$JOINEACH** <変数> <順序付きリスト> "**区切り文字**"\$  
<繰り返し記述>  
**\$END\$**

# 実例

```

246 NL$,
247
248
249 オ ID番号の最大値,
250 _kernel_tmax_semaphore = (TMIN_SEMID + TNUM_SEMID - 1);NL$,
251
252
253 オ 初期化ブロックの生成,
254 TH(SEM.ID_LIST)$,
255 const SEMINIB _kernel_seminib_table[TNUM_SEMID] = {NL$,
256 $JOINEACH semid SEM.ID_LIST ",Wn"$,
257 // sematrが ( [TA_TPRI] ) でない場合 (E_RSATR) ,
258 $IF (SEM.SEMATR[semid] & ~TA_TPRI) != 0$,
259 $ERROR SEM.TEXT_LINE[semid]$E_RSATR: $FORMAT(_("ill
260 $END$,
261
262 // (0 <= isemcnt && isemcnt <= maxsem)でない場合 (E_PAR) ,
263 $IF !(0 <= SEM.ISEM CNT[semid] && SEM.ISEM CNT[semid] <= SEM.
264 $ERROR SEM.TEXT_LINE[semid]$E_PAR: $FORMAT(_("too 1
265 $END$,
266
267 // (1 <= maxsem && maxsem <= TMAX_MAXSEM)でない場合 (E_PAR
268 $IF !(1 <= SEM.MAXSEM[semid] && SEM.MAXSEM[semid] <= TMAX_M
269 $ERROR SEM.TEXT_LINE[semid]$E_PAR: $FORMAT(_("illeg
270 $END$,
271
272 // セマフォ初期化ブロック,
273 $TAB$ { ($SEM.SEMATR[semid]$), ($SEM.ISEM CNT[semid]$), ($SEM
274 $END$$NL$,
275 };$NL$,
276 $NL$,
277
278 // セマフォコントロールブロック,
279 SEMCB _kernel_semc_b_table[TNUM_SEMID];NL$,
280
281 TOPPERS_EMPTY_LABEL(const SEMINIB, _kernel_seminib_table);NL$,
282 TOPPERS_EMPTY_LABEL(SEMCB, _kernel_semc_b_table);NL$,
283
284
285
286
287 ト フ ラ グ,
288
289

```

```

29 $TAB$$TAB$},
30 $END$,
31 $NL$,
32 $JOINEACH semid SEM.ID_LIST ",Wn"$,
33 $TAB$$TAB$$SEMID$":{$NL$,
34 $TAB$$TAB$$TAB$Type:"Semaphore",NL$,
35 $TAB$$TAB$$TAB$Attributes":NL$,
36 $TAB$$TAB$$TAB$NL$,
37 $TAB$$TAB$$TAB$id":$SEMID$,NL$,
38 $TAB$$TAB$$TAB$semcnt":$SEM.ISEM CNT[semid]$,
39 $TAB$$TAB$$TAB$NL$,
40 $TAB$$TAB$},
41 $END$,
42 $NL$,
43 $JOINEACH flgid FLG.ID_LIST ",Wn"$,
44

```

コンパイルすると、kernel.resが生成される。

```

"SEM_ID":{
  "Type":"Semaphore",
  "Attributes":{
    {
      "id":1,
      "semcnt":1
    }
  },
  "SERIAL_RCV_SEM1":{
    "Type":"Semaphore",
    "Attributes":{
      {
        "id":2,
        "semcnt":0
      }
    }
  },
  "SERIAL_SND_SEM1":{
    "Type":"Semaphore",
    "Attributes":{
      {
        "id":3,
        "semcnt":1
      }
    }
  }
}

```

## 5.TLVで変換された標準形式ログを確認する。

時間[us]	リソース	イベント
3603286	LOGTASK	state = WAITING
3650934	MAIN TASK	releaseFromWaiting()
3650934	MAIN TASK	state = RUNNABLE
3651478	MAIN TASK	dispatch()
3651478	MAIN TASK	state = RUNNING
3652188	MAIN TASK	enterSVC(act ts,tskid=1.)
3652389	TASKA	activate()
3652389	TASKA	state = RUNNABLE
3652578	MAIN TASK	leaveSVC(act ts,tskid=1.)
3652724	MAIN TASK	enterSVC(act ts,tskid=2.)
3652925	TASKB	activate()
3652925	TASKB	state = RUNNABLE
3653114	MAIN TASK	leaveSVC(act ts,tskid=2.)
3653260	MAIN TASK	enterSVC(act ts,tskid=3.)
3653461	TASKC	activate()
3653461	TASKC	state = RUNNABLE
3653650	MAIN TASK	leaveSVC(act ts,tskid=3.)
3654590	MAIN TASK	wait()
3654590	MAIN TASK	state = WAITING
3654764	TASKC	dispatch()
3654764	TASKC	state = RUNNING
3654929	TASKC	enterSVC(ini sem,semid=1.)
3655153	SEM ID	semcnt = 1
3655129	TASKC	enterSVC(text ts,tskid=1.)
3655472	TASKC	exit()
3655472	TASKC	state = DORMANT
3655625	TASKC	pri = 8
3655775	TASKA	dispatch()
3655775	TASKA	state = RUNNING
3657126	TASKA	wait()
3657126	TASKA	state = WAITING
3657300	TASKB	dispatch()
3657300	TASKB	state = RUNNING
3657465	TASKB	enterSVC(ini sem,semid=1.)
3657832	TASKA	releaseFromWaiting()
3657832	TASKA	state = RUNNABLE
3658062	SEM ID	semcnt = 1
3659038	TASKB	enterSVC(text ts,tskid=1.)
3659307	TASKB	exit()
3659307	TASKB	state = DORMANT
3659460	TASKB	pri = 9
3659610	TASKA	dispatch()
3659610	TASKA	state = RUNNING
3660002	TASKA	wait()
3660002	TASKA	state = WAITING
5553857	LOGTASK	releaseFromWaiting()
5553857	LOGTASK	state = RUNNABLE
5554207	LOGTASK	dispatch()
5554207	LOGTASK	state = RUNNING
5554372	LOGTASK	leaveSVC(dly ts,tskid=1.)
5603485	LOGTASK	enterSVC(dly ts,tskid=10.)
5603832	LOGTASK	wait()
5603832	LOGTASK	state = WAITING
7528023	LOGTASK	releaseFromWaiting()
7528023	LOGTASK	state = RUNNABLE
7528373	LOGTASK	dispatch()
7528373	LOGTASK	state = RUNNING
7528538	LOGTASK	leaveSVC(dly ts,tskid=10.)
7528748	LOGTASK	enterSVC(dly ts,tskid=10.)
7529095	LOGTASK	wait()
7529095	LOGTASK	state = WAITING
9499691	LOGTASK	releaseFromWaiting()
9499691	LOGTASK	state = RUNNABLE

❖変換に成功すると、交通形式ログがファイル化され、このように出力される。

# 今週の作業予定

---

- ❖ 拡張トレースログの標準形式変換ルール作成。(続き)
  - 終わった後に、可視化変換ルール作成。