AIBO Sony – Debugging notes

Laura D'Angelo
Laboratory of Applied Intelligent Systems (AIS-Lab)
http://ais-lab.dsi.unimi.it
Version 1.0 – 25.Maggio.2005

Sistema di debugging AIBO

Sviluppare codice per AIBO con OPENR-SDK non è un'attività agevole al primo impatto, soprattutto per quanto concerne la parte dedicata al debug per rilevare cause di crash improvvisi. E' possibile attivare una modalità di elaborazione in remote processing sulla quale applicare i classici strumenti di debug per codice C++ (come gdb) quando il codice non viene eseguito completamente su AIBO ma anche da un PC remoto. Come ogni recente ambiente di sviluppo questo approcio permette, l'utilizzo di breakpoint e watch durante l'esecuzione del codice.
Per il codice che viene eseguito direttamente su AIBO sono disponibili come primo strumento tre semplici API per la comunicazione su console wireless accessibile in sola lettura durante l'esecuzione. La console wireless è accessibile tramite comando telnet su porta 59000 ed è possibile visualizzarvi lo standard output che viene generato dalle tre API . La prima API è OSYSPRINT (sostituito di prinf del codice C++). OSYSPRINT come il comando printf può essere abbastanza utile per la generazione di un log e successiva visione. La seconda API è OSYSDEBUG; OSYSDEBUG, a differenza di OSYSPRINT, viene stampato solo al verificarsi di un crash di sistema e solo nel caso in cui sia stata specificata una opportuna opzione al compilatore in fase di compilazione del codice (opzione -DOPENR_DEBUG.). Questi due strumenti non sono pienamente affidabili in quanto la console wireless nella fase di shut-down per crash può essere resa inattiva prima che termini l'output di sistema. La terza API è OSYSLOG1, la quale permette la generazione in output di un messaggio formattato con l'indicazione del livello di log per il quale il messaggio viene mandato in console; il livello viene definito dal programmatore come parametro del comando nel codice. In pratica OSYSLOG1 è l'effettivo strumento per la generazione di un log che viene utilizzato (come di consueto per questo tipo di strumenti) con l'intento di poter indicare a console situazioni differenti di notifica. Per AIBO i livelli di log disponibili sono Warning, Informativa e Errore; in genere queste segnalazioni possono essere inserite in casi in cui un errore può essere gestito dal codice senza che avvenga un arresto del sistema. Anche OSYSLOG1 risulta insufficiente a rilevare situazioni di crash della CPU o della memoria

determinate ad esempio da bug nel codice.

Come strumento più significativo, per rilevare situazioni di crash improvviso, AIBO mette a disposizione un file di log, questa volta generato dal sistema in fase di spegnimento e riposto dallo stesso sulla memory stick. Il file e' nominato emon.log nella cartella open-r. Il file non è di lettura intuitiva ma permette in primo luogo di capire quali sono state le cause del crash. Queste sono tipicamente:

Eccezioni nella TLB
Eccezioni di indirizzi
Eccezioni di istruzioni riservate
Eccezioni Floating-point
Eccezioni di Coprocessor inutilizzabile
Errori di Bus

In secondo luogo, nei casi più comuni di eccezioni nella TLB e di indirizzi errati, è possibile effettuare il parsing del file emon.log per capire in quale oggetto, e specificatamente in quale funzione, l'eccezione si è verificata. Lo script che esegue il parsing si trova nella directory di installazione dei sample di AIBO, sotto il sample Cash cartella util. Questo script tramite l'esecuzione automatizzata dei comandi disponibili del compilatore mipsel esegue una rilevazione dell'indirizzo dell'oggetto in esecuzione al momento dell'errore e calcola l'indirizzo in esecuzione con il quale è possibile risalire alla linea di comando assembly della funzione (routine) in corso.
Nel dettaglio per conoscere l'oggetto che ha causato il cash è sufficiente chiamare lo script (emonLogParser) passandogli come parametro il percorso del file emon.log generato dal sistema:

```
./emonLogParser emon.log
output:
context:        80295f80
object:         <fileOggetto>
```

Successivamente è possibile ripetere l'esecuzione dello script passando come secondo parametro il file immagine generato dal compilatore mipsel dell'oggetto in questione:

```
 ./emonLogParser emon.log <fileOggetto>.nosnap.elf
```

output:

```
context:         80295f80
object:          expressionFinder
badvaddr:        24613090
epc:             22b06bf0
ra:              22b0a298
target address: 22b06bf0 (epc)
gp:              22b88ab0
_gp:             00488ae0
static addr:     00406c20
symbol:          00406bfc T Image::setPixel(unsigned char, int, int)
```

In questo modo si ottenere la funzione e la riga di codice assembly in cui l'eccezzione si e' verificata. Date queste informazioni e' possibile visionare la suddetta riga di codice tramite il comando:

```
<posizione_comando_mipsel>/mipsel-linux-objdump -d -C
/<fileOggetto>.nosnap.elf >
```

Il comando produce un output testuale ridirezionabile su file contente il codice assembler e gli indirizzi relativi; lo static address dovrebbe essere quello in cui si e' verificata l'eccezzione. Lo script emonlogParser non fa altro che eseguire i comandi mipsel disponibili per queste operazioni di ricerca di errore.
Per ottenere un efficacia ancora maggiore sull'ultimo passaggio, oltre a rilevare la funzione e la riga assembler in cui si è verificato il problema, è possibile intercettare anche la riga di codice sorgente , per fare questo e' necessario che il codice in esecuzione sia stato compilato con l'opzione -g e arrivati al comando mipsel-linux-objdump cambiare l'opzione -d in -S, l'output assembler sarà così intervallato dal codice sorgente.
Un'ultima nota in relazione al debug del sistema in casi di crash è relativa al file emon.cfg (presente sulla memory stick di AIBO), file che e' possibile configurare affinché generi una serie di output disponibili al momento del crash; alcune opzioni di configurazione sono:

exception: visualizza le informazioni relative all'oggetto che ha causato l'eccezione e il contenuto dei registri generali.
Tlb: visualizza il contenuto della tabella tlb

objs: visualizza gli oggetti in esecuzione al momento dell'eccezione
stack: visualizza le informazioni di stack dell'oggetto che ha causato l'eccezione.

Maggiori dettagli in merito ai metodi indicati di monitoraggio del sistema in caso di crash sono disponibili sulla documentazione AIBO.