EUROGAM Project
Short Note
on
VxWorks implementation

Edition 0.1
September 1990

. . .

.

CENBG BORDEAUX : P. Kadionik, M.M. Aléonard
CRN STRASBOURG : B. Humbert

————-

Ce document représente le compte-rendu de l'installation et de l'évaluation de VxWorks au CRN de Strasbourg pendant le mois d'août 1990 [1]†.

————-

1 Introduction à VxWorks :

1.1 Considérations générales :

VxWorks est un produit logiciel qui permet de réaliser des applications Temps Réel (TR). VxWorks bénéficie de tous les avantages qu'offre UNIX pendant la phase de développement.
Ainsi, la phase de développement de l'application (édition du programme source, compilation, "make"...) est réalisée sous UNIX. Le code exécutable généré est ensuite téléchargé via le réseau Ethernet dans le chãssis cible pour y faire l'édition de liens, le chargement mémoire puis est exécuté en temps réel.

	UNIX	réseau Ethernet	VxWorks
MISE AU POINT	Développement Debugging	⟺	Application TR
PRODUIT FINI	Application non TR	⟺	Application TR

Des communications entre une tãche UNIX et une tãche VxWorks sont possibles.

1.2 Les possibilités de VxWorks :

La partie réseau a été particulièrement soignée. VxWorks est compatible avec NFS (Network File System) de Sun (partie client seulement). On peut, par conséquent, avoir accès au système de fichiers Sun au niveau du chãssis cible VxWorks.
On peut aussi utiliser rsh et ftp si NFS n'est pas utilisé. NFS peut en effet, ẽtre sélectionné ou non en modifiant les fichiers de configuration (le répertoire des fichiers de configuration étant .../vw/config/).

---

[1] † Le lecteur connaissant déjà VxWorks pourra passer directement au chapĩtre 3.

Les RPC (Remote Procedure Call) ont été implantés sous VxWorks. Les RPC permettent à une tâche sur une machine d'appeler une procédure qui est exécutée par une autre tâche sur la même machine ou sur une machine distante. Signalons que les RPC de VxWorks sont en fait les RPC Sun, et sont donc par la même, 100
Le système d'E/S est compatible C du point de vue du programmeur. On retrouve les primitives classiques : creat(), delete(), open(), close(), read(), write(), ioctl(). On retrouve aussi les routines utilisant les tampons mémoires beaucoup plus performantes (fread(), fwrite()...).

Les drivers sont nombreux. On peut distinguer :
- les "tty drivers" (on peut attacher une console au chãssis cible).
- les "network drivers".
- les "pipe drivers" (communication inter-tâches).
- les "RAM disk drivers".
- les "pseudo-tty drivers".
- les drivers de disque dur. On attend d'ailleurs 3 drivers pour les 3 boĩtiers standards SCSI pour la prochaine version de VxWorks courant septembre 1990.

L'architecture du système de fichiers est basée sur la structure de fichiers RT11.

Des primitives de type timex() (system V) sont aussi utilisables pour réaliser des mesures de temps.

Le mode multi-tâches. La communication inter-tâches. VxWorks est multi- tâches (mono utilisateur). On utilise des niveaux de priorité de 0 à 255 (255 étant le niveau de priorité le plus faible).
Les sémaphores sont également implantés pour régler les problèmes d'accès à une même ressource.
D'autre part, pour la communication inter-tâches, on retrouve des primitives comme pipe(), socket(), signal() bien connues sous UNIX.

2 La force de VxWorks :

L'environnement UNIX utilisé pendant la phase de développement et de debugging est un point fort de VxWorks.

La partie réseau a été particulièrement développée et soignée.

La programmation sous VxWorks se fait en C standard en utilisant en plus la bibliothèque spécifique à VxWorks. La programmation se fait en écrivant des fonctions qui seront exécutées sous VxWorks en les appelant directement par leur nom (il n'y a pas de main() sous VxWorks !!!). Il faut seulement faire attention à la nature du code généré lors de la compilation qui doit rester compatible avec la nature du code exécutable par la CPU cible ! Le code que l'on a généré est téléchargé en mémoire RAM (l'emplacement de ce code en mémoire est géré par VxWorks) via le réseau Ethernet puis on l'exécute en temps réel.

VxWorks est avant tout basé sur l'utilisation d'une seule et unique Table des Symboles qui règle tous les problèmes de référence de symboles externes.

Outre les symboles propres à VxWorks, les symboles définis dans une application sont rajoutés automatiquement à la Table des Symboles lors du chargement de l'application dans le châssis cible (commande *ld < filename.o*), et ainsi sont accessibles à n'importe quelle tâche ultérieurement. On comprend aussi pourquoi pendant la phase de développement d'une application, l'on ne fasse pas appel au linker !

Un seul regret :

Comme sous UNIX, la documentation de VxWorks est vraiment trop pauvre en exemples...

3 Mise en oeuvre de VxWorks :

VxWorks est livré sous forme d'un jeu d'EPROM, d'une cassette contenant le source ainsi que de la documentation afférente.

Les EPROM sont spécifiques à la cible (ex : MVME 147) car elles tiennent compte notamment de la configuration hard de la carte.

Les EPROM remplacent celles du debugger de la 147. Il faut aussi reconfigurer des jumpers de la carte pour tenir compte de la taille mémoire des EPROM. Il faut installer sur le disque de la station de développement le contenu de la cassette VxWorks, puis déclarer sous /etc/hosts le nom du chãssis cible.

Au niveau du chãssis cible, il faut déclarer le nom de la station de développement. Ensuite, le noyau VxWorks ainsi que la Table des Symboles sont chargés en RAM de la carte cible via le réseau Ethernet. Cette phase d'installation et de configuration est très bien décrite dans la documentation VxWorks.

4 Travail réalisé au CRN :

4.1 Matériel disponible :

Nous avions à disposition comme matériel :
- un chãssis VME.
- un chãssis VXI.
- la carte prototype Resource Manager.
- une carte CPU MVME 147 "promée" avec VxWorks.
- une carte d'adaptation MVME 712.
- une station Sun pour le développement.
- une liaison Ethernet entre le Sun et le chãssis VME.

4.2 Acquisition sous VxWorks :

Une carte d'interface de lecture de codeurs (réf CRN) a été implantée dans le chãssis VME cible. Les données issues d'un codeur numérisant le signal d'un générateur d'impulsions (physiques) sont envoyées sur la carte d'interface.
Pour l'acquisition, les programmes suivants ont été écrits :
- initialisation de la carte d'interface (configuration des registres).
- incrémentation d'un histogramme en mémoire RAM.
- mise à zéro d'une zone mémoire.

La carte d'interface a été initialisée (en s'assurant que l'on n'était pas dans l'espace mémoire système). Un "RAM disk" sous VxWorks a été créé (par la commande *ramMkfs*) servant d'intermédiaire entre la station de développement Sun et le chãssis cible. Cet espace mémoire sert de stockage de l'histogramme
Le logiciel de visualisation de spectre sous UNIX que l'on avait à diposition ne permettait de visualiser qu'un fichier de spectre sous un certain format (présence d'un header dans le fichier spectre). Pour ne pas rester à ce niveau, on a rappatrié en zone RAM disk un fichier spectre utilisable par le logiciel de visualisation. Conservant le header du fichier, le reste du fichier en RAM disk est mis à zéro. C'est à cet emplacement mémoire que l'on construira le spectre par incrémentation à chaque événement.
L'acquisition est alors lancée...
On transfert ensuite ce fichier de données vers la station Sun pour y ẽtre visualisé (à l'aide de la commande *copy* de VxWorks).
Cette méthode un peu "rustique" s'est montrée néanmoins très efficace car le but était avant tout de mettre en place rapidement une acquisition et non de développer la partie graphisme.

4.3 Mise en oeuvre des RPC :

La mise en oeuvre des RPC sous Sun et sous VxWorks fera l'objet d'un autre article ("Using Sun RPC", CENBG-FRANCE, P. Kadionik, october 1990).
Signalons que la valeur du time-out des RPC est accessible au programmeur.

4.4 Tests de la carte prototype Resource Manager :

4.4.1 Le Resource Manager et la norme VXI :

Rappelons d'abord, que l'on peut insérer dans un chãssis VXI, 12 cartes VXI, en plus de celle du slot 0. On peut avoir au total 256 devices VXI. Un device VXI réalise une fonction particulière (conversion A/N par exemple).
Pendant la phase dite d'autoconfiguration dynamique qui doit ẽtre exécutée immédiatement après mise sous tension, le Resource Manager va affecter un numéro logique de 0 à 255 aux différents devices VXI contenus dans le chãssis VXI.
A ce numéro logique correspond une adresse de base VME, le numéro logique 0 ayant une adresse de base $FFFFC000 (short I/O). A chaque device VXI sont associés des registres VXI répartis sur 64 octets, soit 32 registres de 16 bits.
les devices VXI sont soit :
- Dynamic Configured : ce que l'on vient de voir et qui est la solution retenue pour EUROGAM.
- Switch Configured : l'adresse logique du device VXI, donc l'adresse physique, est déjà imposée par un système de switchs.

Nous avons ensuite testé la carte prototype Resource Manager qui sera utilisée dans les chãssis VXI pour EUROGAM.

4.4.2 Tests de la carte Resource Manager :

Nous avons couplé la carte prototype Resource Manager à notre carte MVME147 promée avec VxWorks. Nous avons installé le tout dans le chãssis VXI (nous n'avions pas d'autres cartes VXI dans le chãssis...).
Nous avons remarqué que le slot 0 n'est pas accessible par le bus VME, car pour EUROGAM, le Resource Manager a été noyé dans le slot 0 (la norme VXI le permet néanmoins). Le Resource Manager ne permet pas un accès octet. L'accès est ici un accès mot. La norme VXI recommande un accès octet (recommandation C-2-1) mais l'accès mot est permis.

L'accès en lecture ainsi qu'en écriture aux différents registres (en accord avec la norme VXI) de la carte Resource Manager s'est fait sans problème.

Nous pouvons dire, en conclusion de ces tests, que la carte prototype Resource Manager proposée par les ingénieurs anglais, nous a donné pleinement satisfaction.

4.4.3 Mise en place d'un logiciel de tests de cartes VXI :

Un serveur de RPC dont le rõle est de lire le contenu des registres VXI d'un device VXI ou d'y écrire aussi des données a été installé dans le chãssis VXI. Ce logiciel respecte la norme VXI au niveau de l'adressage mémoire (voir ci-dessus).

Un logiciel fonctionnant sur la station Sun a été écrit pour réaliser les appels RPC et traiter les résultats issus des différentes opérations de Lecture/Ecriture des registres VXI d'un device VXI.
Ceci a ainsi permis de mettre en place un outil très précieux pour l'électronicien qui lui permettra de tester et d'initialiser ses cartes VXI.
Cet outil, d'utilisation très aisée, permet de connaître le contenu des 32 registres VXI d'un device. Il permet d'écrire une donnée (mot) dans un registre VXI d'un device. Il permet aussi de savoir, si le registre considéré est un registre ayant deux fonctions distinctes (la valeur écrite est différente de la valeur lue ensuite).

4.5 VxWorks et la carte THEMIS :

Nous avons installé VxWorks (EPROM) sur une carte CPU 68030 (THEMIS). Nous avons fait quelques tests. Les performances de la carte THEMIS sont comparables à celles de la carte MVME147 (vitesse d'exécution...). Signalons un petit bug au niveau d'un fichier de configuration VxWorks pour la carte THEMIS, dans lequel on initialise le registre de cache comme pour un processeur 68020 et non comme pour un processeur 68030 ; d' où une perte de 20% en rapidité de la carte sous VxWorks.

4.6 Mise en PROM d'une application VxWorks :

Ces facilités sont incluses dans le package.
- Il est possible de réaliser d'autres EPROM VxWorks (moniteur) pour d'autres cibles du même type (ici pour la carte MVME 147).

- Il est possible d'insérer dans les EPROM une application (par exemple notre serveur VXI) et son activation.

- Enfin, on peut rajouter le noyau VxWorks ainsi que la Table des Symboles des symboles propres à VxWorks, en plus, en PROM (chargés par défaut en mémoire RAM de la carte cible via Ethernet). On a alors un système complétement autonome et prẽt à la mise sous tension (système embarcable). Remarquons que la Table des Symboles promée contiendra les symboles propres à VxWorks ainsi que les symboles de l'application. Cette dernière possibilité est en cours de réalisation...

Enfin, signalons que l'on peut supprimer des options lors de la génération du fichier S-record (suppression de l'option NETWORK dans les fichiers de configuration) si l'application ne nécessite pas d'accès au réseau Ethernet.

5 Conclusion :

L'installation et la mise en oeuvre de VxWorks est très facile pour une personne connaissant UNIX et le langage C. Mise à part le manque d'exemples dans la documentation, VxWorks nous a pleinement satisfait.

Une configuration minimale pour mettre en place une cellule de développement de logiciels sous VxWorks requiert :
- un chãssis VME.
- une carte CPU MVME147 équipée des EPROM VxWorks.
- une carte d'interface pour la connexion Ethernet à installer dans le chãssis cible.
- une station Sun pour le développement des logiciels.
- un cãble Ethernet.

Au niveau du travail à accomplir, il reste à :
- réaliser une acquisition où la partie visualisation serait active ("alive spectrum").
- tester le mode multi-tãches et les communications inter-tãches.
- mettre en place l'autoconfiguration VXI.
- améliorer l'outil de tests des cartes VXI.
- avec la nouvelle version de VxWorks qui doit arriver courant septembre 1990, explorer le driver SCSI.

VxWorks est tout à fait adapté pour implanter les logiciels "Crate Server" dans les chãssis VXI et VME dans le contexte de réseau de l'acquisition EUROGAM. Son utilisation pour créer le Tape Server est à examiner de près que le driver SCSI aura été étudié.
Pour les cibles VXI "standalone" (pour des tests en labo), une version minimale pourrait ẽtre mise au point. Ceci suppose que tout le logiciel

de test mis au point soit accessible, par exemple, en incluant un disque dur (utilité du driver SCSI de VxWorks) au chãssis cible. Mais, notons qu'il est souhaitable de ne pas se séparer de la station Sun, la console de contrõle attachée au chãssis cible ne servant en fait, que pour l'installation (saisie des différents paramètres : adresse IP,...).