

L4.4: Compte-rendu Essais de Certification Distribution Linux Temps-Reel



| Document rédigé par : | Approuvé par : | Approuvé par : | Approuvé par : |
|---|----------------|----------------|----------------|
| Claude Guilmain (Sagemcom) Olivier Gallot (Axupteam) | | | |



Compte-rendu Essais de Certification Distribution Linux Temps-Réel
Liste des évolutions

| Edition | Date | Evolutions |
|----------------|-------------|-------------------|
| Edition 01 | 16 Mai 2011 | Création |
| | | |
| | | |
| | | |

RTEL4I_Linux_Temps_Reel

Fiche de rapport de test

Projet: RTEL4I Linux Temps Réel

Auteur : Claude GUILMAIN

Imprimé par TestLink le 27/05/2011

2009 © TestLink Community

Table des matières

Plate-forme: SODIMM2410

1.1. Audit_et_Statistiques

mesure de charge et temps processeur

chronogramme

1.2. Traitement_de_Flux

garantir un débit

sauvegarde des caractéristiques des taches temps réel prioritaires

Consommation à date précise

IPC

skin psos

skin vrtx

1.3. Driver_dans_Espace_Utilisateur

Compte-rendu Essais de Certification Distribution Linux Temps-Réel

ULDD

1.4. Acquisition_USB

usb

1.5. Signal_sur_un_GPIO

Stabilité du Signal

Signal carré

1.6. Timer_Haute_Resolution

résolution en nanosencondes

1.7. Asservissement

action réaction

1.8. Temps_Reel_et_Reseau

migration domaine primaire xenomai vers domaine secondaire

pile réseau Linux

pile réseau RTNet

1.9. Documentation

Documentation des API temps réel

Documentation détaillée du générateur

1.10. Allocateur de mémoire dynamique O(1)

allocation mémoire TLSF

1.11. Ordonnanceur

algorithmes EDF et RM

1.12. Prémption

Section critique préemptible

Gestionnaire d'interruption préemptible

Sémaphore à héritage de priorité



Compte-rendu Essais de Certification Distribution Linux Temps-Réel
PTHREAD_PRIO_INHERIT

Compte-rendu Essais de Certification Distribution Linux Temps-Réel

1.1. Test Suite : Audit_et_Statistiques

Cette suite de tests décrit comment les équipes produit vont collecter les informations de configuration et de statistiques

| Cas de Tests RTEL4I-LTR-7: mesure de charge et temps processeur | | |
|---|--|--|
| <u>Auteur :</u> | Olivier GALLOT | |
| <u>Résumé:</u> | Mesurer le temps d'occupation processeur des taches au sein du système et produire un indicateur de charge. | |
| <u>Preconditions:</u> | L'application de traitement de flux est celle de la suite de tests temps réel albatros. | |
| <u>#:</u> | <u>Step actions:</u> | <u>Résultats attendus:</u> |
| 1 | L'utilisateur embarque le module htop dans le firmware cible. Il exécute l'application de traitement de flux puis le programme htop une fois le flux établi. | L'utilisateur visualise dans une meme vue le temps processeur consommé par chaque thread (du domaine primaire xenomai ou secondaire Linux) en exécution dans le système. L'outil présente les consommations instantanées et moyennes pour chaque thread et pour l'ensemble pour produire l'indicateur de charge système. |
| <u>Dernier résultat:</u> | Echec | |
| Session | Essais de certification de la Distribution Linux Temps-Réel Version 1 | |
| Tester | Olivier GALLOT | |
| Notes d'exécution | Les modules atop et htop ne sont pas intégrés dans le générateur de firmware buildroot. Les utilitaires du package/procps ne permettent pas de suivre chacun des pthreads embarqués dans une application multi-threads. | |
| <u>Exigences fonctionnelles</u> | EX_ 11: Pouvoir énumérer les taches temps reel EX_ 12: Pouvoir connaitre la priorité d'une tache EX_ 13: Pouvoir connaitre l'état d'une tache EX_ 14: Pouvoir savoir de quel processus Linux dépend une tache EX_ 15: Pouvoir mesurer la charge de chacune des taches temps réel EX_ 16: Pouvoir mesurer la charge de l'ensemble des taches temps réel EX_ 23: Pouvoir visualiser les suivis temporel de la charge de chaque tache EX_ 24: Pouvoir visualiser les suivis temporel de l'ensemble des | |

**Compte-rendu Essais de Certification Distribution Linux Temps-Réel**

| | |
|-------------------|---------|
| | charges |
| <u>mots clés:</u> | Aucun |

Compte-rendu Essais de Certification Distribution Linux Temps-Réel

| Cas de Tests RTEL4I-LTR-8: chronogramme | | |
|---|--|--|
| <u>Auteur :</u> | Olivier GALLOT | |
| <u>Résumé:</u> | | |
| mise en oeuvre de LTT | | |
| <u>Preconditions:</u> | | |
| L'application de traitement de flux est celle de la suite de tests temps réel albatros. | | |
| <u>#:</u> | <u>Step actions:</u> | <u>Résultats attendus:</u> |
| 1 | <p>L'utilisateur procède comme suit:</p> <ul style="list-style-type: none"> • choisi, télécharge et applique les patches adeos et LTTng correspondants à la version du noyau Linux utilisée. • applique le patch trace_mark channel event au patch de xenomai • applique le patch xenomai modifié ci-dessus au noyau Linux • compilation du noyau Linux patché • mkinitramfs • compilation des modules noyau LTTng • construction de ltt-control et de lttv <p>Une fois le firmware généré et chargé sur la cible, l'utilisateur configure le traceur LTTng puis exécute l'application de traitement de flux.</p> | <p>L'utilisateur visualise dans une seule vue les chronogrammes de l'ensemble des threads (temps réel ou non) en exécution dans le système. Les événements et les appels systèmes sont représentés. L'exécution dans les espaces utilisateur et noyau sont différenciés.</p> |
| <u>Dernier résultat:</u> Echec | | |
| <u>Session</u> | Essais de certification de la Distribution Linux Temps-Réel Version 1 | |



Compte-rendu Essais de Certification Distribution Linux Temps-Réel

| | |
|---------------------------------|---|
| Tester | Olivier GALLOT |
| Notes d'exécution | Le patch et le package LTT pour la version noyau 2.6.25 ne sont pas intégrés dans le générateur buildroot. |
| <u>Exigences fonctionnelles</u> | <p>EX_ 11: Pouvoir énumérer les taches temps réel</p> <p>EX_ 12: Pouvoir connaître la priorité d'une tache</p> <p>EX_ 13: Pouvoir connaître l'état d'une tache</p> <p>EX_ 14: Pouvoir savoir de quel processus Linux dépend une tache</p> <p>EX_ 20: Pouvoir mesurer la gigue moyenne au reveil de chacune des taches (variance/écart-type)</p> <p>EX_ 21: Pouvoir mesurer le retard négatif moyen de chacune des taches périodiques (variance/ecart-type)</p> <p>EX_ 22: Pouvoir visualiser la chronologie des évènements</p> <p>EX_ 41: Pouvoir tracer l'exécution d'Adeos I pipe</p> <p>EX_ 42: Pouvoir tracer le code de xenomai avec LTT.</p> <p>EX_ 58: Pouvoir mesurer le temps de changement de contexte. (lmbench)</p> |
| <u>mots clés:</u> | Aucun |

Compte-rendu Essais de Certification Distribution Linux Temps-Réel

1.2. Test Suite : Traitement_de_Flux

Suite de tests qui met en œuvre les modèles de traitement de flux et de stress CPU

Le traitement d'un flux de données est réalisé dans l'espace utilisateur par l'enchaînement des tâches de production, d'encodage, de décodage et enfin de consommation des données. Les paquets du flux de données sont consommés aux dates programmées par la tâche de production. Les tâches sont exécutées dans l'espace temps réel xenomai.

| Cas de Tests RTEL4I-LTR-2: garantir un débit | | |
|--|--|--|
| <u>Auteur :</u> | Olivier GALLOT | |
| <u>Résumé:</u> | | |
| Le débit de données traitées mesuré dans l'espace temps réel est équivalent à celui obtenu dans l'espace non temps réel. | | |
| <u>Preconditions:</u> | | |
| L'application de traitement de flux est celle de la suite de tests temps réel albatros. | | |
| <u>#:</u> | <u>Step actions:</u> | <u>Résultats attendus:</u> |
| 1 | <p>L'utilisateur exécute l'application de traitement de flux implémentée suivant l'API POSIX dans l'espace temps réel primaire Xenomai puis relève les débits et latences de traitement obtenus.</p> <p>L'utilisateur compile cette même application POSIX pour l'exécuter dans Linux non temps réel.</p> | <p>Les débits et latences de traitement obtenus pour l'exécution de l'application dans Linux non temps réel ne sont pas meilleurs que ceux obtenus dans l'espace temps réel xenomai.</p> |
| <u>Dernier résultat:</u> Echec | | |
| Session | Essais de certification de la Distribution Linux Temps-Réel Version 1 | |
| Tester | Olivier GALLOT | |
| Notes d'exécution | <p>L'utilisateur lance l'application de traitement de flux dans le domaine Linux avec le paramétrage du test xenomai RTEL4I-LTR-4 "Consommation à date précise".</p> <p>La latence pour traiter un paquet est plus petite dans le domaine Linux. La différence de latence est en moyenne d'environ 230 microsecondes.</p> <p>Lorsque le système n'est pas chargé, le débit de traitement (traversée du pipeline) est donc plus faible pour la solution temps réel xenomai.</p> | |

Compte-rendu Essais de Certification Distribution Linux Temps-Réel

| | |
|---------------------------------|---|
| | <pre># linux_thread_stream -w 1000 -t 8000000 -u 8000000 -n 10 -l 100000000 -m 10000 000 -r 10 will send 10 packets per production period and expect 10 packets received per consumer period initialization done (consumption,underrun,lost): (1000,2,0) , conso (mean,stddev,jitter): (0,0,0) , process (mean,stddev,jitter): (118132297,31999251,112118016) (consumption,underrun,lost): (1000,0,0) , conso (mean,stddev,jitter): (0,0,0) , process (mean,stddev,jitter): (118150092,28223344,10553856) (consumption,underrun,lost): (1000,0,0) , conso (mean,stddev,jitter): (0,0,0) , process (mean,stddev,jitter): (118149904,28224130,10635776) (consumption,underrun,lost): (1000,0,0) , conso (mean,stddev,jitter): (0,0,0) , process (mean,stddev,jitter): (118158627,28187586,10564864) (consumption,underrun,lost): (1000,0,0) , conso (mean,stddev,jitter): (0,0,0) , process (mean,stddev,jitter): (118161474,28175649,10797312) (consumption,underrun,lost): (1000,0,0) , conso (mean,stddev,jitter): (0,0,0) , process (mean,stddev,jitter): (118176073,28114348,10638848) (consumption,underrun,lost): (1000,0,0) , conso (mean,stddev,jitter): (0,0,0) , process (mean,stddev,jitter): (118162944,28169483,10557184) (consumption,underrun,lost): (1000,0,0) , conso (mean,stddev,jitter): (0,0,0) , process (mean,stddev,jitter): (118155231,28201819,10744064) (consumption,underrun,lost): (1000,0,0) , conso (mean,stddev,jitter): (0,0,0) , process (mean,stddev,jitter): (118159044,28185837,10846208) (consumption,underrun,lost): (1000,0,0) , conso (mean,stddev,jitter): (0,0,0) , process (mean,stddev,jitter): (118150979,28219630,10536960) (consumption,underrun,lost): (1000,0,0) , conso (mean,stddev,jitter): (0,0,0) , process (mean,stddev,jitter): (118149034,28227772,10664960) (consumption,underrun,lost): (1000,0,0) , conso (mean,stddev,jitter): (0,0,0) , process (mean,stddev,jitter): (118150330,28222347,10555392) (consumption,underrun,lost): (1000,0,0) , conso (mean,stddev,jitter): (0,0,0) , process (mean,stddev,jitter): (118147421,28234523,10737920) (consumption,underrun,lost): (1000,0,0) , conso (mean,stddev,jitter): (0,0,0) , process (mean,stddev,jitter): (118152964,28211316,10641920) (consumption,underrun,lost): (1000,0,0) , conso (mean,stddev,jitter): (0,0,0) , process (mean,stddev,jitter): (118147469,28234323,10537984) (consumption,underrun,lost): (1000,0,0) , conso (mean,stddev,jitter): (0,0,0) , process (mean,stddev,jitter): (118147090,28235909,10651904) (consumption,underrun,lost): (1000,0,0) , conso (mean,stddev,jitter): (0,0,0) , process (mean,stddev,jitter): (118148754,28228944,10562048)</pre> |
| <u>Exigences fonctionnelles</u> | EX_ 1: Pouvoir facilement mettre en Suivre le temps réel. EX_ 2: Disposer d'un système robuste EX_ 3: Ne pas perdre les performances en introduisant le temps réel. |
| <u>mots clés:</u> | Aucun |

Compte-rendu Essais de Certification Distribution Linux Temps-Réel

Cas de Tests RTEL4I-LTR-3: sauvegarde des caractéristiques des taches temps réel prioritaires

Auteur : Olivier GALLOT

Résumé:

Génération d'un stress temps réel et contraintes temps réel respectées pour les taches temps réel prioritaires.

Preconditions:

L'application de traitement de flux est celle de la suite de tests temps réel albatros.

| <u>#:</u> | <u>Step actions:</u> | <u>Résultats attendus:</u> |
|-----------|---|--|
| 1 | <p>L'utilisateur fixe les priorités des taches de l'application de traitement de flux puis mesure le débit du flux obtenu et relève la gigue de traitement d'un paquet.</p> <p>Dans un second temps, l'utilisateur exécute en concurrence de l'application de traitement de flux une tache de stress système dans l'espace temps réel primaire xenomai avec une priorité plus faible que celles des taches du traitement de flux.</p> | <p>Que la tache de stress temps réel de faible priorité soit exécutée ou non, le débit du flux et la gigue de traitement obtenus sont égaux.</p> |

Dernier résultat: **Bloqué**

Session Essais de certification de la Distribution Linux Temps-Réel Version 1

Tester Olivier GALLOT

Notes d'exécution

L'application de traitement de flux utilise 4 pthreads.
L'application de stress temps réel 2 pthreads.

Avec la skin posix Xenomai, l'appel à la fonction pthread_create induit la consommation de 8 mega octet de mémoire dynamique (malloc).
Par exemple la commande pmap sur le test clocktest fourni dans le package xenomai renvoi l'infomation suivante:
pmap `pidof clocktest`
401c6000 8188K rw--- [anon]

Par conséquent l'exécution en parallèle de l'application de traitement de flux et de l'application de stress temps réel n'est pas possible. Sur les 64 mega octets disponible sur la carte sodimm2410 la mémoire consommée par nos deux tests est donc trop importante. Le noyau finit



Compte-rendu Essais de Certification Distribution Linux Temps-Réel

| | |
|---------------------------------|--|
| | par tuer l'application de traitement de flux. |
| <u>Exigences fonctionnelles</u> | EX_4: Conserver les propriétés temporelles d'un système lors de l'ajout d'une application temps réel si la capacité de la machine le permet. |
| <u>mots clés:</u> | Aucun |

Compte-rendu Essais de Certification Distribution Linux Temps-Réel

| Cas de Tests RTEL4I-LTR-4: Consommation à date précise | | |
|---|--|--|
| <u>Auteur :</u> | Olivier GALLOT | |
| <u>Résumé:</u> | | |
| Les paquets de données du flux sont consommés sans retard. | | |
| <u>Preconditions:</u> | | |
| L'application de traitement de flux est celle de la suite de tests temps réel albatros. | | |
| <u>#:</u> | <u>Step actions:</u> | <u>Résultats attendus:</u> |
| 1 | L'utilisateur exécute l'application de traitement de flux dans l'espace temps réel xenomai. | La gigue de traitement d'un paquet de données du flux n'excède pas 50 nanosecondes. |
| 2 | L'utilisateur réalise une session de debugage conjointe de l'application avec gdb et du noyau avec kgdb. | L'application xenomai et le noyau sont debugable bien que le temps réel soit "cassé". |
| 3 | L'utilisateur recompile le noyau avec l'option "Detect Soft Lockups" activée puis exécute l'application de traitement de flux. | La détection de problème de synchronisation et de prise de verrou au sein du noyau s'applique également au code xenomai. |
| <u>Dernier résultat:</u> Reussi | | |
| Session | Essais de certification de la Distribution Linux Temps-Réel Version 1 | |
| Tester | Olivier GALLOT | |
| Notes d'exécution | <p>L'application de traitement de flux s'exécute sur la carte sodimm2410 avec la commande ci-après.</p> <p>Les paquets sont consommés sans retards, le temps de traversée du pipeline de taches est en moyenne de 11,838 ms avec une gigue de 8,07 ms à 8,12 ms</p> <p>Les exigences EX_35 (vxworks), EX_43 (kgdb) ne sont pas courvertes.</p> <pre># signal_stream -w 1000 -t 8000000 -u 8000000 -n 10 -l 100000000 -m 10000000 -r 10 (consumption,underrun,lost,full): (1000,0,0,0) , conso (mean,stddev,jitter): (0,0,0) , process (mean,stddev,jitter): (118385390,27219390,8061183) , waiting (mean,stddev,jitter): () (consumption,underrun,lost,full): (1000,0,0,0) , conso (mean,stddev,jitter): (0,0,0) , process (mean,stddev,jitter):</pre> | |



Compte-rendu Essais de Certification Distribution Linux Temps-Réel

```
(118385583,27218552,8053610) , waiting (mean,stddev,jitter): ()
(consumption,underrun,lost,full): (1000,0,0,0) , conso
(mean,stddev,jitter): (0,0,0) , process (mean,stddev,jitter):
(118385536,27218755,8095740) , waiting (mean,stddev,jitter): ()
(consumption,underrun,lost,full): (1000,0,0,0) , conso
(mean,stddev,jitter): (0,0,0) , process (mean,stddev,jitter):
(118385296,27219800,8061657) , waiting (mean,stddev,jitter): ()
(consumption,underrun,lost,full): (1000,0,0,0) , conso
(mean,stddev,jitter): (0,0,0) , process (mean,stddev,jitter):
(118385127,27220536,8108402) , waiting (mean,stddev,jitter): ()
(consumption,underrun,lost,full): (1000,0,0,0) , conso
(mean,stddev,jitter): (0,0,0) , process (mean,stddev,jitter):
(118385385,27219413,8107573) , waiting (mean,stddev,jitter): ()
(consumption,underrun,lost,full): (1000,0,0,0) , conso
(mean,stddev,jitter): (0,0,0) , process (mean,stddev,jitter):
(118385258,27219964,8082249) , waiting (mean,stddev,jitter): ()
(consumption,underrun,lost,full): (1000,0,0,0) , conso
(mean,stddev,jitter): (0,0,0) , process (mean,stddev,jitter):
(118385411,27219300,8071361) , waiting (mean,stddev,jitter): ()
(consumption,underrun,lost,full): (1000,0,0,0) , conso
(mean,stddev,jitter): (0,0,0) , process (mean,stddev,jitter):
(118385393,27219378,8060001) , waiting (mean,stddev,jitter): ()
(consumption,underrun,lost,full): (1000,0,0,0) , conso
(mean,stddev,jitter): (0,0,0) , process (mean,stddev,jitter):
(118385291,27219821,8095266) , waiting (mean,stddev,jitter): ()
(consumption,underrun,lost,full): (1000,0,0,0) , conso
(mean,stddev,jitter): (0,0,0) , process (mean,stddev,jitter):
(118385512,27218859,8088520) , waiting (mean,stddev,jitter): ()
(consumption,underrun,lost,full): (1000,0,0,0) , conso
(mean,stddev,jitter): (0,0,0) , process (mean,stddev,jitter):
(118385246,27220017,8071479) , waiting (mean,stddev,jitter): ()
(consumption,underrun,lost,full): (1000,0,0,0) , conso
(mean,stddev,jitter): (0,0,0) , process (mean,stddev,jitter):
(118385231,27220083,8114438) , waiting (mean,stddev,jitter): ()
(consumption,underrun,lost,full): (1000,0,0,0) , conso
(mean,stddev,jitter): (0,0,0) , process (mean,stddev,jitter):
(118385150,27220434,8071834) , waiting (mean,stddev,jitter): ()
(consumption,underrun,lost,full): (1000,0,0,0) , conso
(mean,stddev,jitter): (0,0,0) , process (mean,stddev,jitter):
(118385394,27219373,8053728) , waiting (mean,stddev,jitter): ()
(consumption,underrun,lost,full): (1000,0,0,0) , conso
(mean,stddev,jitter): (0,0,0) , process (mean,stddev,jitter):
(118385525,27218804,8057278) , waiting (mean,stddev,jitter): ()
(consumption,underrun,lost,full): (1000,0,0,0) , conso
(mean,stddev,jitter): (0,0,0) , process (mean,stddev,jitter):
(118384551,27223041,8081302) , waiting (mean,stddev,jitter): ()
(consumption,underrun,lost,full): (1000,0,0,0) , conso
(mean,stddev,jitter): (0,0,0) , process (mean,stddev,jitter):
(118384043,27225250,8078343) , waiting (mean,stddev,jitter): ()
(consumption,underrun,lost,full): (1000,0,0,0) , conso
```



Compte-rendu Essais de Certification Distribution Linux Temps-Réel

| | |
|---------------------------------|--|
| | (mean,stddev,jitter): (0,0,0) , process (mean,stddev,jitter): (118384032,27225296,8059290) , waiting (mean,stddev,jitter): () (consumption,underrun,lost,full): (1000,0,0,0) , conso (mean,stddev,jitter): (0,0,0) , process (mean,stddev,jitter): (118384042,27225253,8121657) , waiting (mean,stddev,jitter): () |
| <u>Exigences fonctionnelles</u> | <p>EX_ 29: Disposer du temps réel dur dans le monde user</p> <p>EX_ 34: Pouvoir utiliser la skin POSIX de Xenomai sur les plateformes de références.</p> <p>EX_ 35: Pouvoir utiliser la skin VxWorks de Xenomai sur les plateformes de références.</p> <p>EX_ 43: Pouvoir debugger les taches xenomai (espace utilisateur, espace noyau), (gdb,kgdb)</p> <p>EX_ 47: Disposer de timers POSIX pour les plateformes cibles de références avec une précision de l'ordre de quelques nano secondes.</p> <p>EX_ 5: Pouvoir piloter une application par le temps</p> <p>EX_ 50: Pourvoir gérer des horloges avec la glibc ou la uclibc</p> <p>EX_ 52: Pouvoir utiliser des bibliothèques partagées sur des systèmes sans mmu avec la glibc ou la uclibc.</p> <p>EX_ 55: Pouvoir compiler les bibliothèques partagées pour XIP (Execute In Place) sur les plateformes de références avec la glibc ou la uclibc.</p> <p>EX_ 56: Pouvoir utiliser la détection de problème de synchronisation au sein du noyau et analyser la prise de verrou (deadlock, lockdep).</p> <p>EX_ 58: Pouvoir mesurer le temps de changement de contexte. (lmbench)</p> |
| <u>mots clés:</u> | Aucun |

Compte-rendu Essais de Certification Distribution Linux Temps-Réel

| Cas de Tests RTEL4I-LTR-10: IPC | | |
|---------------------------------|---|---|
| <u>Auteur :</u> | Olivier GALLOT | |
| <u>Résumé:</u> | Le traitement de flux est réparti sur plusieurs processus qui échangent leurs données avec les IPC. | |
| <u>Preconditions:</u> | L'application de traitement de flux est celle de la suite de tests temps réel albatros. | |
| <u>#:</u> | <u>Step actions:</u> | <u>Résultats attendus:</u> |
| 1 | L'utilisateur exécute dans le domaine primaire xenomai l'ensemble des processus qui instancient les taches du traitement de flux. Les buffers de paquets sont des mémoires partagées entre ces processus et les fifos sont des pipes nommés. L'utilisateur règle un débit de traitement D, le calcul des statistiques donne la moyenne et la gigue de la latence de traitement d'un paquet de données. Dans un second temps l'utilisateur exécute une application de stress système. | Les processus du domaine primaire xenomai utilisent les IPC et malgré l'application du stress système la latence de traitement d'un paquet reste constante. |
| <u>Dernier résultat:</u> | Non Disponible | |
| Session | Essais de certification de la Distribution Linux Temps-Réel Version 1 | |
| Tester | Claude GUILMAIN g107219 | |
| Notes d'exécution | En attente de la création du programme qui permet de réaliser ce test | |
| <u>Exigences fonctionnelles</u> | EX_ 30: Pouvoir faire communiquer les taches xenomai avec des processus linux au travers de fifo. EX_ 31: Pouvoir faire communiquer les taches xenomai avec des processus linux au travers de mémoires partagées. EX_ 51: Pouvoir utiliser des mémoires partagées avec la glibc ou la uclibc EX_ 59: Pouvoir mesurer la performance des IPC.(lmbench) | |
| <u>mots clés:</u> | Aucun | |

Compte-rendu Essais de Certification Distribution Linux Temps-Réel

Cas de Tests RTEL4I-LTR-17: skin psos

Auteur : Olivier GALLOT

Résumé:

Porter sur Linux une application psos temps réel
L'utilisateur fait le portage d'une application psos vers xenomai avec la skin psos

Preconditions:

L'objet du portage peut être l'application de traitement de flux de la suite de tests temps réel albatros.

| <u>#:</u> | <u>Step actions:</u> | <u>Résultats attendus:</u> |
|-----------|---|--|
| 1 | L'utilisateur compile et embarque dans le firmware cible l'application de traitement de flux utilisant l'API de la skin psos xenomai. | Le binaire de l'application de traitement de flux est construit sans erreur. L'application s'exécute aussi bien que celle construite avec l'API POSIX. |

Dernier résultat: **Non Disponible**

Session Essais de certification de la Distribution Linux Temps-Réel Version 1

Tester Claude GUILMAIN g107219

Notes d'exécution En attente de la création du programme qui permet de réaliser ce test

Exigences fonctionnelles EX_36: Pouvoir utiliser la skin PSOS de Xenomai sur les plateformes de références.

mots clés: Aucun

Compte-rendu Essais de Certification Distribution Linux Temps-Réel

Cas de Tests RTEL4I-LTR-18: skin vrtx

Auteur : Olivier GALLOT

Résumé:

Porter sur Linux une application VRTX temps réel.

L'utilisateur fait le portage d'une application vrtx vers xenomai avec la skin VRTX

Préconditions:

L'objet du portage peut être l'application de traitement de flux de la suite de tests temps réel albatros.

| <u>#:</u> | <u>Step actions:</u> | <u>Résultats attendus:</u> |
|-----------|---|--|
| 1 | L'utilisateur compile et embarque dans le firmware cible l'application de traitement de flux utilisant l'API de la skin vrtx xenomai. | Le binaire de l'application de traitement de flux est construit sans erreur. L'application s'exécute aussi bien que celle construite avec l'API POSIX. |

Dernier résultat: **Non Disponible**

Session Essais de certification de la Distribution Linux Temps-Réel Version 1

Tester Claude GUILMAIN g107219

Notes d'exécution En attente de la création du programme qui permet de réaliser ce test

Exigences fonctionnelles EX_37: Pouvoir utiliser la skin VRTX de Xenomai sur les plateformes de références.

mots clés: Aucun

1.3. Test Suite : Driver_dans_Espace_Utilisateur

Le but des tests est d'exécuter un driver temps réel Xenomai dans l'espace utilisateur sans pénalité sur la réception des événements

| Cas de Tests RTEL4I-LTR-9: ULDD | | |
|--|---|---|
| <u>Auteur :</u> | Olivier GALLOT | |
| <u>Résumé:</u> | Exécuter un driver temps réel dans l'espace utilisateur. | |
| <u>#:</u> | <u>Step actions:</u> | <u>Résultats attendus:</u> |
| 1 | <p>Le firmware embarque un module noyau générique uio qui relaye l'interruption de l'UART du noyau au gestionnaire d'interruption de l'UART dans l'espace utilisateur. Une tâche temps réel extrait depuis l'espace utilisateur les caractères de la fifo de l'UART. L'utilisateur mesure la latence entre le dépôt d'un caractère sur l'UART et son extraction par la tâche temps réel exécutée dans l'espace utilisateur.</p> <p>Dans un second temps, l'utilisateur exécute l'application de stress système.</p> | Malgré la charge système appliquée, la latence mesurée est constante. |
| <u>Dernier résultat:</u> | Non Disponible | |
| <u>Session</u> | Essais de certification de la Distribution Linux Temps-Réel Version 1 | |
| <u>Tester</u> | Claude GUILMAIN g107219 | |
| <u>Notes d'exécution</u> | En attente de la création du programme qui permet de réaliser ce test | |
| <u>Exigences fonctionnelles</u> | EX_28: Pouvoir traiter une interruption dans une fonction de l'espace utilisateur. | |
| <u>mots clés:</u> | Aucun | |

Compte-rendu Essais de Certification Distribution Linux Temps-Réel

1.4. Test Suite : Acquisition_USB

But des tests : Lire les paquets présentés sur un lien USB High Speed avec une latence inférieure à 100 microsecondes

| Cas de Tests RTEL4I-LTR-11: usb | | |
|---------------------------------|---|---|
| <u>Auteur :</u> | Olivier GALLOT | |
| <u>Résumé:</u> | Utiliser un lien usb 2.0 en temps réel. | |
| <u>#:</u> | <u>Step actions:</u> | <u>Résultats attendus:</u> |
| 1 | <p>Une tache temps réel de l'espace utilisateur reçoit des paquets de données sur un lien usb 2.0 piloté par la couche noyau usb4rt. L'utilisateur mesure la latence entre le dépôt sur le lien usb et le retour de la fonction de lecture d'un paquet usb.</p> <p>Dans un second temps l'utilisateur exécute l'application de stress système en sus de la lecture des paquets usb.</p> | Malgré la charge système appliquée la latence pour disposer d'un paquet usb dans la tache temps réel de l'espace utilisateur est constante. |
| <u>Dernier résultat:</u> | Non Disponible | |
| Session | Essais de certification de la Distribution Linux Temps-Réel Version 1 | |
| Tester | Olivier GALLOT | |
| Notes d'exécution | La stack usb temps réel usb4rt n'est pas disponible dans le générateur de firmware. | |
| <u>Exigences fonctionnelles</u> | <p>EX_ 40: Pouvoir utiliser un contrôleur usb (uhci et ehci) en temps réel (usb4rt,xenomai).</p> <p>EX_ 49: Pouvoir utiliser les fonctions i/o asynchrones avec la glibc ou la uclibc</p> | |
| <u>mots clés:</u> | Aucun | |

1.5. Test Suite : Signal_sur_un_GPIO

But des tests : Emettre un signal cadencé sur une fréquence précise

| Cas de Tests RTEL4I-LTR-26: Stabilité du Signal | | |
|---|--|---|
| <u>Auteur :</u> | Claude GUILMAIN | |
| <u>Résumé:</u> | | |
| <p>On génère une trace périodique avec un GPIO (0/1) et on l'affiche sur un oscilloscope.</p> <p>Le but de ce test est de montrer la non influence d'un transfert de données sur la gestion des process au niveau du scheduling. Ce test va démontrer la stabilité du temps réel apporté par Xenomai.</p> | | |
| <u>#:</u> | <u>Step actions:</u> | <u>Résultats attendus:</u> |
| 1 | <p>Génération d'une trace périodique avec un GPIO (0/1).</p> <p>Création d'un programme avec un compteur de période 5ms (ajustable)</p> <p>Affichage de la trace générée par ce programme sur un oscilloscope.</p> <p>Le programme qui génère la trace est soit sous Linux (rt_square) donc non temps-réel, soit sous Xenomai (Xenomai_square) et basé sur l'API POSIX.</p> <p>On effectue simultanément un gros transfert réseau vers la carte (scp par exemple).</p> | <p>Dans le cas de la trace sous linux on doit constater du jitter au niveau de l'affichage sur l'oscilloscope (décrochage de l'affichage).</p> <p><u>Dans le cas de la trace sous Xenomai, le signal généré doit rester stable.</u></p> <p>En conclusion Xenomai apporte une gestion plus rigoureuse du temps réel que n'a pas une distribution Linux de base.</p> |
| 2 | | |
| <u>Dernier résultat:</u> | Reussi | |
| <u>Session</u> | Essais de certification de la Distribution Linux Temps-Réel Version 1 | |



Compte-rendu Essais de Certification Distribution Linux Temps-Réel

| | |
|-------------------|--|
| Tester | Olivier GALLOT |
| Notes d'exécution | <p>La charge appliquée au système est la suivante: + Un transfert de plusieurs giga octets d'un équipement distant vers /dev/null de la carte + <code>http://weather.ou.edu/~apw/projects/stress/stress-1.0.1.tar.gz</code> exécuté avec la commande <code>stress --io 4 --vm 2 --vm-bytes 128K --timeout 500s --cpu 120 &</code> Cette commande est entrée autant de fois que nécessaire pour atteindre une charge CPU de 98 à 99%.</p> <p>La capture à l'oscilloscope du signal GPIO est réalisée dans les 3 cas suivants:</p> <ol style="list-style-type: none"> 1. test dans Linux avec le scheduler par défaut <code>rt_square_gpio -p 10000000</code> 2. test dans Linux avec le scheduler <code>SCHED_FIFO rt_square_gpio -p 10000000 -r</code> 3. test dans Xenomai <p>Sans application de charge système l'oscilloscope capture un beau signal carré.</p> <p>Avec application des charges décrites ci dessus l'oscilloscope capture:</p> <ol style="list-style-type: none"> 1. un signal carré très déformé dans le cas de Linux avec scheduler par défaut => gigue supérieure à la période de 10 ms 2. un signal carré déformé dans le cas de Linux avec scheduler <code>SCHED_FIFO</code> => gigue jusqu'a 4,5 ms pour la période 10 ms 3. un signal carré intact dans le cas de Xenomai => gigue de 8 microsecondes au plus <p>Traces du programme dans les cas 2 et 3.</p> <p>2. RT Linux</p> <pre> Loop= 10200 sec= 1859705012 nsec= 356118000 delta= 12936000 ns Loop= 10400 sec= 1859705014 nsec= 353122000 delta= 9555000 ns Loop= 10600 sec= 1859705016 nsec= 353188000 delta= 10007000 ns Loop= 10800 sec= 1859705018 nsec= 353180000 delta= 9988000 ns Loop= 11000 sec= 1859705020 nsec= 353819000 delta= 10634000 ns Loop= 11200 sec= 1859705022 nsec= 353119000 delta= 9934000 ns Loop= 11400 sec= 1859705024 nsec= 353164000 delta= 9991000 ns Loop= 11600 sec= 1859705026 nsec= 354117000 delta= 10925000 ns Loop= 11800 sec= 1859705028 nsec= 354461000 delta= 11278000 ns Loop= 12000 sec= 1859705030 nsec= 353161000 delta= 9776000 ns Loop= 12200 sec= 1859705032 nsec= 357711000 delta= 14512000 ns Loop= 12400 sec= 1859705034 nsec= 353192000 delta= 9915000 ns Loop= 12600 sec= 1859705036 nsec= 354982000 delta= 11861000 ns Loop= 12800 sec= 1859705038 nsec= 353216000 delta= 10051000 ns Loop= 13000 sec= 1859705040 nsec= 353189000 delta= 10009000 ns Loop= 13200 sec= 1859705042 nsec= 353182000 delta= 6910000 ns Loop= 13400 sec= 1859705044 nsec= 355485000 delta= 12300000 ns Loop= 13600 sec= 1859705046 nsec= 357280000 delta= 14162000 ns </pre> |

Compte-rendu Essais de Certification Distribution Linux Temps-Réel

| | |
|--|--|
| | <p>Loop= 13800 sec= 1859705048 nsec= 353183000 delta= 10004000 ns Loop= 14000 sec= 1859705050 nsec= 353159000 delta= 9774000 ns Loop= 14200 sec= 1859705052 nsec= 353476000 delta= 10310000 ns Loop= 14400 sec= 1859705054 nsec= 354127000 delta= 10938000 ns</p> <p>3. Xenomai</p> <p>Loop= 365800 sec= 4589 nsec= 820714727 delta= 10046982 ns Loop= 366000 sec= 4590 nsec= 820669165 delta= 9999053 ns Loop= 366200 sec= 4591 nsec= 820666798 delta= 9999882 ns Loop= 366400 sec= 4592 nsec= 820668928 delta= 9999527 ns Loop= 366600 sec= 4593 nsec= 820689638 delta= 10024852 ns Loop= 366800 sec= 4594 nsec= 820668336 delta= 9999763 ns Loop= 367000 sec= 4595 nsec= 820666798 delta= 9998935 ns Loop= 367200 sec= 4596 nsec= 820668218 delta= 10001539 ns Loop= 367400 sec= 4597 nsec= 820663721 delta= 9995740 ns Loop= 367600 sec= 4598 nsec= 820668691 delta= 10003669 ns Loop= 367800 sec= 4599 nsec= 820667508 delta= 9999764 ns Loop= 368000 sec= 4600 nsec= 820667389 delta= 10000828 ns Loop= 368200 sec= 4601 nsec= 820669993 delta= 10000592 ns Loop= 368400 sec= 4602 nsec= 820662892 delta= 9995858 ns Loop= 368600 sec= 4603 nsec= 820667626 delta= 10000355 ns Loop= 368800 sec= 4604 nsec= 820667152 delta= 9999171 ns Loop= 369000 sec= 4605 nsec= 820676265 delta= 10000710 ns Loop= 369200 sec= 4606 nsec= 820675436 delta= 10001420 ns Loop= 369400 sec= 4607 nsec= 820665969 delta= 9995385 ns Loop= 369600 sec= 4608 nsec= 820668335 delta= 9996923 ns Loop= 369800 sec= 4609 nsec= 820670229 delta= 10001775 ns Loop= 370000 sec= 4610 nsec= 820667980 delta= 9999171 ns Loop= 370200 sec= 4611 nsec= 820671649 delta= 9999763 ns Loop= 370400 sec= 4612 nsec= 820664548 delta= 9995029 ns Loop= 370600 sec= 4613 nsec= 820667152 delta= 9998225 ns Loop= 370800 sec= 4614 nsec= 820668217 delta= 10000119 ns Loop= 371000 sec= 4615 nsec= 820672359 delta= 10002959 ns Loop= 371200 sec= 4616 nsec= 820668098 delta= 9992544 ns Loop= 371400 sec= 4617 nsec= 820667033 delta= 9999053 ns</p> |
| <p><u>Exigences fonctionnelles</u></p> | <p>EX_ 1: Pouvoir facilement mettre en Suivre le temps réel. EX_ 2: Disposer d'un système robuste EX_ 25: Pouvoir ajouter simplement le co-noyau temps réel xenomai au noyau linux.</p> |
| <p><u>mots clés:</u></p> | <p>Aucun</p> |

Compte-rendu Essais de Certification Distribution Linux Temps-Réel

| Cas de Tests RTEL4I-LTR-1: Signal carré | | |
|--|---|--|
| <u>Auteur :</u> | Olivier GALLOT | |
| <u>Résumé:</u> | | |
| Générer un signal sur un GPIO à une fréquence précise. | | |
| <u>#:</u> | <u>Step actions:</u> | <u>Résultats attendus:</u> |
| 1 | <p>Une tâche temps réel du domaine utilisateur programme un timer avec une période de 10ms. Pour chaque début de période la tâche temps réel écrit un GPIO lui même relié à un oscilloscope.</p> <p>Dans un second temps l'utilisateur applique une charge système.</p> | Malgré l'application de la charge système l'oscilloscope enregistre un signal carré. |
| <u>Dernier résultat:</u> Reussi | | |
| Session | Essais de certification de la Distribution Linux Temps-Réel Version 1 | |
| Tester | Olivier GALLOT | |
| Notes d'exécution | <p>La charge appliquée au système est la suivante: + Un transfert de plusieurs giga octets d'un équipement distant vers /dev/null de la carte + <code>http://weather.ou.edu/~apw/projects/stress/stress-1.0.1.tar.gz</code> exécuté avec la commande <code>stress --io 4 --vm 2 --vm-bytes 128K --timeout 500s --cpu 120 &</code> Cette commande est entrée autant de fois que nécessaire pour atteindre une charge CPU de 98 à 99%.</p> <p>La capture à l'oscilloscope du signal GPIO est réalisée dans les 3 cas suivants:</p> <ol style="list-style-type: none"> 1. test dans Linux avec le scheduler par défaut <code>rt_square_gpio -p 10000000</code> 2. test dans Linux avec le scheduler <code>SCHED_FIFO</code> <code>rt_square_gpio -p 10000000 -r</code> 3. test dans Xenomai <p>Sans application de charge système l'oscilloscope capture un beau signal carré.</p> <p>Avec application des charges décrites ci dessus l'oscilloscope capture:</p> <ol style="list-style-type: none"> 1. un signal carré très déformé dans le cas de Linux avec scheduler par défaut => gigue supérieure à la période de 10 ms 2. un signal carré déformé dans le cas de Linux avec scheduler <code>SCHED_FIFO</code> => gigue jusqu'a 4,5 ms pour la période 10 ms | |



Compte-rendu Essais de Certification Distribution Linux Temps-Réel

3. un signal carré intact dans le cas de Xenomai => gigue de 8 microsecondes au plus

Traces du programme dans les cas 2 et 3.

2. RT Linux

```

Loop= 10200 sec= 1859705012 nsec= 356118000 delta= 12936000 ns
Loop= 10400 sec= 1859705014 nsec= 353122000 delta= 9555000 ns
Loop= 10600 sec= 1859705016 nsec= 353188000 delta= 10007000 ns
Loop= 10800 sec= 1859705018 nsec= 353180000 delta= 9988000 ns
Loop= 11000 sec= 1859705020 nsec= 353819000 delta= 10634000 ns
Loop= 11200 sec= 1859705022 nsec= 353119000 delta= 9934000 ns
Loop= 11400 sec= 1859705024 nsec= 353164000 delta= 9991000 ns
Loop= 11600 sec= 1859705026 nsec= 354117000 delta= 10925000 ns
Loop= 11800 sec= 1859705028 nsec= 354461000 delta= 11278000 ns
Loop= 12000 sec= 1859705030 nsec= 353161000 delta= 9776000 ns
Loop= 12200 sec= 1859705032 nsec= 357711000 delta= 14512000 ns
Loop= 12400 sec= 1859705034 nsec= 353192000 delta= 9915000 ns
Loop= 12600 sec= 1859705036 nsec= 354982000 delta= 11861000 ns
Loop= 12800 sec= 1859705038 nsec= 353216000 delta= 10051000 ns
Loop= 13000 sec= 1859705040 nsec= 353189000 delta= 10009000 ns
Loop= 13200 sec= 1859705042 nsec= 353182000 delta= 6910000 ns
Loop= 13400 sec= 1859705044 nsec= 355485000 delta= 12300000 ns
Loop= 13600 sec= 1859705046 nsec= 357280000 delta= 14162000 ns
Loop= 13800 sec= 1859705048 nsec= 353183000 delta= 10004000 ns
Loop= 14000 sec= 1859705050 nsec= 353159000 delta= 9774000 ns
Loop= 14200 sec= 1859705052 nsec= 353476000 delta= 10310000 ns
Loop= 14400 sec= 1859705054 nsec= 354127000 delta= 10938000 ns

```

3. Xenomai

```

Loop= 365800 sec= 4589 nsec= 820714727 delta= 10046982 ns
Loop= 366000 sec= 4590 nsec= 820669165 delta= 9999053 ns
Loop= 366200 sec= 4591 nsec= 820666798 delta= 9999882 ns
Loop= 366400 sec= 4592 nsec= 820668928 delta= 9999527 ns
Loop= 366600 sec= 4593 nsec= 820689638 delta= 10024852 ns
Loop= 366800 sec= 4594 nsec= 820668336 delta= 9999763 ns
Loop= 367000 sec= 4595 nsec= 820666798 delta= 9998935 ns
Loop= 367200 sec= 4596 nsec= 820668218 delta= 10001539 ns
Loop= 367400 sec= 4597 nsec= 820663721 delta= 9995740 ns
Loop= 367600 sec= 4598 nsec= 820668691 delta= 10003669 ns
Loop= 367800 sec= 4599 nsec= 820667508 delta= 9999764 ns
Loop= 368000 sec= 4600 nsec= 820667389 delta= 10000828 ns
Loop= 368200 sec= 4601 nsec= 820669993 delta= 10000592 ns
Loop= 368400 sec= 4602 nsec= 820662892 delta= 9995858 ns
Loop= 368600 sec= 4603 nsec= 820667626 delta= 10000355 ns
Loop= 368800 sec= 4604 nsec= 820667152 delta= 9999171 ns
Loop= 369000 sec= 4605 nsec= 820676265 delta= 10000710 ns
Loop= 369200 sec= 4606 nsec= 820675436 delta= 10001420 ns
Loop= 369400 sec= 4607 nsec= 820665969 delta= 9995385 ns
Loop= 369600 sec= 4608 nsec= 820668335 delta= 9996923 ns
Loop= 369800 sec= 4609 nsec= 820670229 delta= 10001775 ns

```



Compte-rendu Essais de Certification Distribution Linux Temps-Réel

| | |
|---------------------------------|---|
| | <p>Loop= 370000 sec= 4610 nsec= 820667980 delta= 9999171 ns Loop= 370200 sec= 4611 nsec= 820671649 delta= 9999763 ns Loop= 370400 sec= 4612 nsec= 820664548 delta= 9995029 ns Loop= 370600 sec= 4613 nsec= 820667152 delta= 9998225 ns Loop= 370800 sec= 4614 nsec= 820668217 delta= 10000119 ns Loop= 371000 sec= 4615 nsec= 820672359 delta= 10002959 ns Loop= 371200 sec= 4616 nsec= 820668098 delta= 9992544 ns Loop= 371400 sec= 4617 nsec= 820667033 delta= 9999053 ns</p> |
| <u>Exigences fonctionnelles</u> | <p>EX_ 1: Pouvoir facilement mettre en Suivre le temps réel. EX_ 2: Disposer d'un système robuste EX_ 25: Pouvoir ajouter simplement le co noyau temps réel xenomai au noyau linux.</p> |
| <u>mots clés:</u> | Aucun |

Compte-rendu Essais de Certification Distribution Linux Temps-Réel

1.6. Test Suite : Timer_Haute_Resolution

But des tests : Planifier des échéances avec une précision de l'ordre de la nano seconde

| Cas de Tests RTEL4I-LTR-6: résolution en nanosecondes | | |
|---|--|---|
| <u>Auteur :</u> | Olivier GALLOT | |
| <u>Résumé:</u> | Planifier des échéances avec un précision de l'ordre de la nanoseconde. | |
| <u>#:</u> | <u>Step actions:</u> | <u>Résultats attendus:</u> |
| 1 | <p>Une tache temps réel du domaine utilisateur programme un timer avec une période définie à la nanoseconde près. Pour chaque début de période la tache temps réel écrit un GPIO lui même relié à un oscilloscope.</p> <p>Dans un second temps l'utilisateur applique une charge système.</p> | L'oscilloscope enregistre un signal périodique. |
| <u>Dernier résultat:</u> | Echec | |
| Session | Essais de certification de la Distribution Linux Temps-Réel Version 1 | |
| Tester | Olivier GALLOT | |
| Notes d'exécution | <p>1.Pour Linux Malgré la configuration du noyau avec l'option CONFIG_HIGH_RES_TIMERS=y le générateur buildroot produit un firmware avec des timers de la résolution du tick système soit 5ms.</p> <p>2.Pour Xenomai # xenomai_square_gpio -p 1000000 Loop= 2000 sec= 844 nsec= 852220028 delta= 1001302 ns wait_period failed: err 110, overruns: 1 Xenomai: POSIX: destroyed thread c3cc0610</p> <p>L'erreur 110 est ETIMEDOUT, qui signifie qu'au moment où pthread_wait_np est appelée l'échéance est déjà passée.</p> | |
| <u>Exigences fonctionnelles</u> | <p>EX_ 50: Pouvoir gérer des horloges avec la glibc ou la uclibc</p> <p>EX_ 7: Disposer de timer avec une précision de l'ordre de la nano seconde</p> <p>EX_ 8: Avoir des latences de l'ordre de la microseconde.</p> | |
| <u>mots clés:</u> | Aucun | |

Compte-rendu Essais de Certification Distribution Linux Temps-Réel

1.7. Test Suite : Asservissement

But des tests : Enchaîner les lectures et écritures de GPIO, dans le cadre d'un asservissement, suivant une échéance.

| Cas de Tests RTEL4I-LTR-5: action réaction | | |
|---|--|---|
| <u>Auteur :</u> | Olivier GALLOT | |
| <u>Résumé:</u> | | |
| Enchaîner la lecture et l'écriture de GPIOs suivant une échéance. | | |
| <u>#:</u> | <u>Step actions:</u> | <u>Résultats attendus:</u> |
| 1 | <p>Un équipement génère un signal sur un GPIO1 qui génère à son tour une interruption afin de notifier l'arrivée de données à une tâche temps réel exécutée dans l'espace utilisateur. L'utilisateur calibre une échéance avant laquelle la tâche temps réel peut écrire sur le GPIO2.</p> <p>Dans un second temps l'utilisateur applique un stress système.</p> | <p>Malgré la charge système la tâche temps réel écrit le GPIO2 avant l'arrivée de l'échéance.</p> |
| <u>Dernier résultat:</u> | Non Disponible | |
| <u>Session</u> | Essais de certification de la Distribution Linux Temps-Réel Version 1 | |
| <u>Tester</u> | Claude GUILMAIN g107219 | |
| <u>Notes d'exécution</u> | En attente de la création du programme qui permet de réaliser ce test | |
| <u>Exigences fonctionnelles</u> | EX_ 6: Pouvoir piloter une application par les événements EX_ 60: Pouvoir tracer le temps passe avec les interruptions désactivées, la préemption désactivée et lors des changements de contexte (CONFIG_SCHED_TRACER). | |
| <u>mots clés:</u> | Aucun | |

1.8. Test Suite : Temps_Réel_et_Réseau

| Cas de Tests RTEL4I-LTR-16: migration domaine primaire xenomai vers domaine secondaire | | |
|--|--|---|
| <u>Auteur :</u> | Olivier GALLOT | |
| <u>Résumé:</u> | | |
| Garder sa priorité face aux autres taches primaires lorsqu'une tache primaire Xenomai effectue un appel système linux. | | |
| Une tache primaire Xenomai fait un appel système qui induit un changement de domaine. Cette tache ne perd pas sa priorité face aux autres taches primaires du système. | | |
| <u>Preconditions:</u> | | |
| L'application qui échange les paquets RTP peut être l'application type VOIP de la suite de tests temps réel albatros. | | |
| <u>#:</u> | <u>Step actions:</u> | <u>Résultats attendus:</u> |
| 1 | <p>Une tache temps réel Xenomai ouvre une socket Linux UDP pour recevoir le flux RTP émis par une machine (type pc) du réseau local. L'attente d'un paquet RTP est réalisée avec l'interface POSIX select() qui induit la migration de la tache primaire vers le domaine secondaire: Linux.</p> <p>Une fois le flux RTP établi, l'utilisateur exécute dans l'espace primaire Xenomai une tache de stress système attribuée d'une plus faible priorité que la tache de réception du flux RTP.</p> | Malgré sa migration dans l'espace secondaire Linux, la tache de réception du flux ne perd pas sa priorité sur la tache de stress. |
| <u>Dernier résultat:</u> Bloqué | | |
| Session | Essais de certification de la Distribution Linux Temps-Réel Version 1 | |
| Tester | Olivier GALLOT | |
| Notes d'exécution | L'utilisateur joue le scénario décrit dans le test RTEL4I-LTR-12 puis ajoute l'exécution du stress temps réel de priorité basse (priorité 2 versus 6 pour rtp_receive). rt_stress 2 500 0 3 0 | |



Compte-rendu Essais de Certification Distribution Linux Temps-Réel

| | |
|--|--|
| | <p>Les statistiques affichées par l'application rtp_receive montre que la dérive de la latence système est sensiblement la même que la tâche de stress temps réel soit exécutée ou pas.</p> <p>L'absence du package LTT dans la distribution RTEL4I ne permet pas de conclure sur le fait que rtp_receive ne perde pas sa priorité face à rt_stress lorsque qu'il migre du domaine primaire Xenomai au domaine secondaire Linux.</p> |
| <p><u>Exigences fonctionnelles</u></p> | <p>EX_32: Pouvoir garder sa priorité face aux autres taches primaires lorsqu'une tâche primaire Xenomai effectue un appel système Linux.</p> |
| <p><u>mots clés:</u></p> | <p>Aucun</p> |

Compte-rendu Essais de Certification Distribution Linux Temps-Réel

| Cas de Tests RTEL4I-LTR-12: pile réseau Linux | | |
|---|---|--|
| <u>Auteur :</u> | Olivier GALLOT | |
| <u>Résumé:</u> | Mise en concurrence du flux de données sur la pile réseau Linux. | |
| <u>Préconditions:</u> | Disponibilité de deux machines puissantes (type pc) et d'un switch ethernet L'application qui échange les paquets RTP peut être l'application type VOIP de la suite de tests temps réel albatros. | |
| <u>#:</u> | <u>Step actions:</u> | <u>Résultats attendus:</u> |
| 1 | La plateforme temps réel est reliée dans un réseau local à deux machines distantes par l'intermédiaire d'un switch ethernet. Elle établit avec des jeux de priorités différents deux flux RTP sur la pile réseau Linux depuis des applications temps réel Xenomai. Chacun des flux est à destination d'une machine distante. Les flux générés sont calibrés afin d'obtenir des débits conséquents. Dans un second temps l'application de stress système est exécutée en sus d'un flux parasite iperf établi entre la plateforme temps réel et une des machines distantes. | Les débits des flux RTP établis sont conservés malgré l'application des charges système et réseau. |
| <u>Dernier résultat:</u> | Bloqué | |
| Session | Essais de certification de la Distribution Linux Temps-Réel Version 1 | |
| Tester | Olivier GALLOT | |
| Notes d'exécution | <pre> root@sodimm2410-xenomai> ptpd -u 192.168.1.109 root@i386-xenomai> ptpd -u 192.168.1.100 root@sodimm2410-xenomai> rtp_receive -i 192.168.1.100/5000 root@i386-xenomai> rtp_send -o 192.168.1.100/5000 </pre> <p>Le temps Linux des machines qui échangent les paquets RTP est donc synchroniser par le démon ptpd.</p> <p>L'application rtp_receive recupère le temps Linux depuis le domaine Xenomai avec l'API <code>__real_clock_gettime(CLOCK_REALTIME, &tp)</code></p> <p>Malheureusement le temps Linux récupéré par cette API est</p> | |



Compte-rendu Essais de Certification Distribution Linux Temps-Réel

| | |
|---------------------------------|---|
| | <p>complètement desynchronisé du temps porté par la socket API SO_TIMESTAMPNS.</p> <p>Le temps de remonté du packet jusqu'au domaine Xenomai espace user n'est donc pas calculable.</p> |
| <u>Exigences fonctionnelles</u> | <p>EX_ 57: Pouvoir tester (benchmarking tools [interbench, netperf, hackbench, dbench, stress]) les latences dans le système, la performance de l'ordonnanceur, mesurer la performance réseau.</p> <p>EX_ 58: Pouvoir mesurer le temps de changement de contexte. (lmbench)</p> |
| <u>mots clés:</u> | Aucun |

Compte-rendu Essais de Certification Distribution Linux Temps-Réel

| Cas de Tests RTEL4I-LTR-13: pile réseau RTNet | | |
|---|---|--|
| <u>Auteur :</u> | Olivier GALLOT | |
| <u>Résumé:</u> | | |
| Mise en concurrence du flux de données sur la pile réseau RTNet. | | |
| <u>Preconditions:</u> | | |
| L'application qui échange les paquets RTP peut être le portage sur RTNet de l'application type VOIP de la suite de tests temps réel Albatros. | | |
| <u>#:</u> | <u>Step actions:</u> | <u>Résultats attendus:</u> |
| 1 | La plateforme temps réel est reliée dans un réseau local à deux machines distantes par l'intermédiaire d'un switch ethernet. Elle établit avec des jeux de priorités différents deux flux RTP sur la pile réseau RTNet depuis des applications temps réel Xenomai. Chacun des flux est à destination d'une machine distante. Les flux générés sont calibrés afin d'obtenir des débits conséquents. Dans un second temps l'application de stress système est exécutée en sus d'un flux parasite iperf établi entre la plateforme temps réel et une des machines distantes. | Les débits des flux RTP établis sont conservés malgré l'application des charges système et réseau. |
| <u>Dernier résultat:</u> Non Disponible | | |
| Session | Essais de certification de la Distribution Linux Temps-Réel Version 1 | |
| Tester | Olivier GALLOT | |
| Notes d'exécution | La pile réseau RTNet n'est pas intégrée au générateur de firmware buildroot. La pile réseau RTNet pour le driver ethernet DM9000 de la carte sodimm2410 n'est pas finalisée. http://www.mail-archive.com/rtnet-users@lists.sourceforge.net/msg03102.html | |
| <u>Exigences fonctionnelles</u> | EX_ 38: Pouvoir utiliser un contrôleur gigabit ethernet en temps réel (rtnet,xenomai). EX_ 57: Pouvoir tester (benchmarking tools [interbench, netperf, hackbench, dbench, stress]) les latences dans le système, la performance de l'ordonnanceur, mesurer la performance réseau. EX_ 58: Pouvoir mesurer le temps de changement de contexte. (lmbench) | |
| <u>mots clés:</u> | Aucun | |

Compte-rendu Essais de Certification Distribution Linux Temps-Réel

1.9. Test Suite : Documentation

Accéder à la documentation sans accès à internet

| Cas de Tests RTEL4I-LTR-14: Documentation des API temps réel | | |
|---|---|---|
| <u>Auteur :</u> | Olivier GALLOT | |
| <u>Résumé:</u> | | |
| Accéder facilement à la documentation des solutions Linux Temps Réel | | |
| L'utilisateur consulte la documentation des API xenomai sur son poste de développement sans connexion à internet. | | |
| Depuis la racine du générateur de firmware l'utilisateur consulte le répertoire doc du module xenomai où des fichiers html et txt sont à disposition. | | |
| <u>#:</u> | <u>Step actions:</u> | <u>Résultats attendus:</u> |
| 1 | L'utilisateur ouvre la documentation des "clocks and timers services" dans son éditeur vi. | La documentation des skins et de l'ensemble des composants xenomai est consultable depuis l'éditeur Vi. |
| 2 | L'utilisateur navigue dans la documentation html afin de trouver les API vxWorks supportées. | La documentation html est accessible hors ligne. |
| <u>Dernier résultat:</u> Reussi | | |
| Session | Essais de certification de la Distribution Linux Temps-Réel Version 1 | |
| Tester | Olivier GALLOT | |
| Notes d'exécution | <p>La documentation hors ligne des api xenomai sont disponibles.</p> <p>Par exemple l'utilisateur peut consulter l'api des clocks dans le fichier output/build/xenomai-2.4.4_OW/doc/generated/html/api/group__clock.html.</p> <p>La documentation au format texte n'inclue pas les api.</p> | |
| <u>Exigences fonctionnelles</u> | EX_ 9: Avoir une documentation détaillée des API de programmation | |
| <u>mots clés:</u> | Aucun | |



Compte-rendu Essais de Certification Distribution Linux Temps-Réel

Compte-rendu Essais de Certification Distribution Linux Temps-Réel
Cas de Tests RTEL4I-LTR-15: Documentation détaillée du générateur

Auteur : Olivier GALLOT

Résumé:

Accéder facilement à la documentation de l'outil de génération du firmware.

L'utilisateur consulte la documentation du générateur de firmware sur son poste de développement sans connexion à internet.

| <u>#:</u> | <u>Step actions:</u> | <u>Résultats attendus:</u> |
|-----------|--|--|
| 1 | L'utilisateur ouvre dans son éditeur Vi un fichier texte du répertoire doc à la racine du générateur de firmware. | La documentation du générateur de firmware est disponible au format texte. |
| 2 | L'utilisateur ouvre dans son navigateur les fichiers de documentation au format html | La documentation au format html est accessible hors ligne. |
| 3 | L'utilisateur consulte la documentation et trouve facilement comment insérer un nouveau module dans le firmware cible. | <p>La documentation décrit comment:</p> <ul style="list-style-type: none"> • insérer un nouveau module • définir les dépendances avec les autres modules • où trouver les binaires construits |

Dernier résultat: **Reussi**

Session Essais de certification de la Distribution Linux Temps-Réel Version 1

Tester Olivier GALLOT

Notes d'exécution

L'utilisateur peut consulter hors ligne une partie de l'architecture du générateur buildroot en parcourant le fichier docs/buildroot.html.

Néanmoins la documentation des modules CMake disponible sur internet ne l'est pas dans les fichiers du répertoire docs.

La documentation au format texte disponible docs/README est très incomplète.

La documentation est celle de buildroot et inclue donc aucune documentation sur le plan de version, les profils d'un firmware ou

**Compte-rendu Essais de Certification Distribution Linux Temps-Réel**

| | |
|---------------------------------|--|
| | simplement quelles options activer pour réaliser une session de debug du firmware. |
| <u>Exigences fonctionnelles</u> | EX_10: Avoir une documentation détaillée des outils de configuration et d'administration |
| <u>mots clés:</u> | Aucun |

1.10. Test Suite : Allocateur de mémoire dynamique O(1)

Prévoir le temps nécessaire pour allouer de la mémoire dynamique.

| Cas de Tests RTEL4I-LTR-19: allocation mémoire TLSF | | |
|--|---|---|
| <u>Auteur :</u> | Olivier GALLOT | |
| <u>Résumé:</u> | | |
| Disposer d'un allocateur de mémoire dynamique approprié au temps réel. Pouvoir prévoir le temps nécessaire pour allouer de la mémoire dynamique. La fonction malloc() est implémentée avec l'algorithme tlsf de complexité O(1). | | |
| <u>Preconditions:</u> | | |
| disponibilité du patch tlsf de xenomai | | |
| <u>#:</u> | <u>Step actions:</u> | <u>Résultats attendus:</u> |
| 1 | Depuis l'interface du générateur de firmware, l'utilisateur sélectionne l'application du patch tlsf de l'allocateur de mémoire dynamique de xenomai puis construit le firmware cible. | Le firmware cible embarque l'allocateur dynamique xenomai tlsf. |
| <u>Dernier résultat:</u> Non Disponible | | |
| Session | Essais de certification de la Distribution Linux Temps-Réel Version 1 | |
| Tester | Olivier GALLOT | |
| Notes d'exécution | Le patch tlsf de xenomai n'est pas disponible pour la plateforme. | |
| <u>Exigences fonctionnelles</u> | EX_27: Pouvoir utiliser un algorithme d'allocation mémoire de complexité O(1). (tlsf memory allocator). | |
| <u>mots clés:</u> | Aucun | |

1.11. Test Suite : Ordonnanceur

Choisir l'algorithme d'ordonnancement temps réel.

| Cas de Tests RTEL4I-LTR-20: algorithmes EDF et RM | | |
|--|---|---|
| <u>Auteur :</u> | Olivier GALLOT | |
| <u>Résumé:</u> | Choisir l'algorithme d'ordonnancement temps réel entre EDF (Early Dead line First) et Rate Monotic. | |
| <u>Pré-conditions:</u> | Disponibilité des algorithmes EDF et RM pour Xenomai | |
| <u>#:</u> | <u>Step actions:</u> | <u>Résultats attendus:</u> |
| 1 | Depuis l'interface du générateur de firmware, l'utilisateur sélectionne pour le module xenomai l'algorithme d'ordonnancement EDF (Early Dead line First). | Xenomai, embarqué dans le firmware cible, ordonnance les taches temps réel suivant l'algorithme EDF. |
| 2 | Depuis l'interface du générateur de firmware, l'utilisateur sélectionne pour le module xenomai l'algorithme d'ordonnancement Rate Monotonic. | Xenomai, embarqué dans le firmware cible, ordonnance les taches temps réel suivant l'algorithme Rate Monotonic. |
| <u>Dernier résultat:</u> | Non Disponible | |
| Session | Essais de certification de la Distribution Linux Temps-Réel Version 1 | |
| Tester | Olivier GALLOT | |
| Notes d'exécution | L'utilisateur ne peut pas choisir l'algorithme d'ordonnancement de Xenomai. | |
| <u>Exigences fonctionnelles</u> | EX_26: Pouvoir choisir l'algorithme d'ordonnancement (EDF, RM (Rate Monothonic)) | |
| <u>mots clés:</u> | Aucun | |

1.12. Test Suite : Prémption

Rendre préemptible la majeure partie du code du noyau.

| Cas de Tests RTEL4I-LTR-21: Section critique préemptible | | |
|---|---|--|
| <u>Auteur :</u> | Olivier GALLOT | |
| <u>Résumé:</u> | Pouvoir décider qu'une section critique du noyau soit préemptible ou non. | |
| <u>#:</u> | <u>Step actions:</u> | <u>Résultats attendus:</u> |
| 1 | Depuis l'interface du générateur de firmware, l'utilisateur sélectionne l'application du patch PREEMPT_RT. | Les taches temps réel qui définissent des sections critiques avec des verrous d'API spinlock peuvent s'endormir. |
| <u>Dernier résultat:</u> | Echec | |
| Session | Essais de certification de la Distribution Linux Temps-Réel Version 1 | |
| Tester | Olivier GALLOT | |
| Notes d'exécution | Le patch PREEMPT_RT pour la plateforme sodimm2410 n'est pas intégré dans le générateur buildroot. | |
| <u>Exigences fonctionnelles</u> | EX_44: Pouvoir décider qu'une section critique du code noyau soit préemptible ou non (CONFIG_PREEMPT_RT spinlock_t, rwlock_t, rawspinlock_t). | |
| <u>mots clés:</u> | Aucun | |

Compte-rendu Essais de Certification Distribution Linux Temps-Réel

Cas de Tests RTEL4I-LTR-22: Gestionnaire d'interruption préemptible

Auteur : Olivier GALLOT

Résumé:

Pouvoir préempter un gestionnaire d'interruption.

| <u>#:</u> | <u>Step actions:</u> | <u>Résultats attendus:</u> |
|-----------|---|--|
| 1 | Depuis l'interface du générateur de firmware, l'utilisateur sélectionne l'application du patch PREEMPT_RT | Un gestionnaire d'interruption est préemptible pour exécuter une tâche ou un gestionnaire d'interruption plus prioritaire. |

Dernier résultat: **Echec**

Session Essais de certification de la Distribution Linux Temps-Réel Version 1

Tester Olivier GALLOT

Notes d'exécution Le patch PREEMPT_RT pour la plateforme sodimm2410 n'est pas intégré dans le générateur buildroot.

Exigences fonctionnelles EX_46: Pouvoir préempter les handlers d'interruption (CONFIG_PREEMPT_RT,isr as preemptible kernel thread).

mots clés: Aucun

Compte-rendu Essais de Certification Distribution Linux Temps-Réel
Cas de Tests RTEL4I-LTR-23: Sémaphore à héritage de priorité

Auteur : Olivier GALLOT

Résumé:

Eviter les inversion de priorité.
Hériter de la priorité de la tâche qui attend la libération d'un verrou.

| <u>#:</u> | <u>Step actions:</u> | <u>Résultats attendus:</u> |
|-----------|--|--|
| 1 | Depuis l'interface du générateur de firmware, l'utilisateur sélectionne l'application du patch PREEMPT_RT pour la configuration d'un noyau entièrement préemptible puis construit le firmware. | Une tâche prioritaire en attente de libération d'un verrou ne peut pas être bloquée par une tâche de moindre priorité détenant ce même verrou. |

Dernier résultat: **Echec**

Session: Essais de certification de la Distribution Linux Temps-Réel Version 1

Tester: Olivier GALLOT

Notes d'exécution: Le patch PREEMPT_RT pour la plateforme sodimm2410 n'est pas intégré dans le générateur buildroot.

Exigences fonctionnelles: EX_45: Pouvoir hériter de la priorité de la tâche qui attend la libération d'un verrou (mutex et sémaphore). (CONFIG_PREEMPT_RT, inversion de priorité).

mots clés: Aucun

Compte-rendu Essais de Certification Distribution Linux Temps-Réel

| Cas de Tests RTEL4I-LTR-24: PTHREAD_PRIO_INHERIT | | |
|--|--|---|
| <u>Auteur :</u> | Olivier GALLOT | |
| <u>Résumé:</u> | | |
| Eviter les inversions de priorité depuis l'espace utilisateur. | | |
| Dans l'espace utilisateur, hériter de la priorité de la tâche qui attend la libération d'un verrou: API pthread_mutexattr_setprotocol (PTHREAD_PRIO_INHERIT) | | |
| <u>Preconditions:</u> | | |
| L'application de traitement de flux est celle de la suite de tests temps réel albatros. | | |
| <u>#:</u> | <u>Step actions:</u> | <u>Résultats attendus:</u> |
| 1 | Dans le code de l'application de traitement de flux, les mutex sont initialisés suivant le protocole PTHREAD_PRIO_INHERIT avec l'API pthread_mutexattr_setprotocol(). | Une tâche détenant un mutex hérite de la priorité d'une tâche concurrente de plus haute priorité en attente sur ce mutex. |
| <u>Dernier résultat:</u> Echec | | |
| <u>Session</u> | Essais de certification de la Distribution Linux Temps-Réel Version 1 | |
| <u>Tester</u> | Olivier GALLOT | |
| <u>Notes d'exécution</u> | <p>L'utilisateur ajoute les codes suivants dans le programme signal_stream.c du module albatros:</p> <pre>+ pthread_mutexattr_t mattr ; + pthread_mutexattr_init (&mattr) ; + pthread_mutexattr_setprotocol(&mattr, PTHREAD_PRIO_INHERIT);</pre> <p>- if (pthread_mutex_init (&time_to_produce_lock , NULL) < 0) { + if (pthread_mutex_init (&time_to_produce_lock , &mattr) < 0) {</p> <p>puis relance la génération du firmware.</p> <p>La compilation échoue avec l'erreur suivante: signal_stream.c:1211:40: error: 'PTHREAD_PRIO_INHERIT' undeclared (first use in this function)</p> <p>En fait PTHREAD_PRIO_INHERIT est défini dans pthread.h mais uniquement dans le cadre defined(__KERNEL__) defined(__XENO_SIM__).</p> | |
| <u>Exigences</u> | EX_ 45: Pouvoir hériter de la priorité de la tâche qui attend la | |



Compte-rendu Essais de Certification Distribution Linux Temps-Réel

| | |
|-----------------------|--|
| <u>fonctionnelles</u> | libération d'un verrou (mutex et sémaphore). (CONFIG_PREEMPT_RT, inversion de priorité). EX_ 48: Pouvoir hériter de la priorité de la tâche qui attend la libération d'un verrou (mutex et sémaphore) dans l'espace utilisateur pour la glibc ou la uclibc (PTHREAD_PRIO_INHERIT). |
| <u>mots clés:</u> | Aucun |

Synthèse des résultats

Résultats par suites de tests de haut niveau

| Suite | Total | Non exécuté | Reussi | Echec | Bloqué | Non Disponible | Inconnu | Achévé [%] |
|--------------------------------------|-------|-------------|--------|-------|--------|----------------|---------|------------|
| Audit_et_Statistiques | 2 | 0 | 0 | 2 | 0 | 0 | 0 | 100.0 |
| Traitement_de_Flux | 6 | 0 | 1 | 1 | 1 | 3 | 0 | 100.0 |
| Driver_dans_Espace_Utilisateur | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 100.0 |
| Acquisition_USB | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 100.0 |
| Signal_sur_un_GPIO | 2 | 0 | 2 | 0 | 0 | 0 | 0 | 100.0 |
| Timer_Haute_Resolution | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 100.0 |
| Asservissement | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 100.0 |
| Temps_Reel_et_Reseau | 3 | 0 | 0 | 0 | 2 | 1 | 0 | 100.0 |
| Documentation | 2 | 0 | 2 | 0 | 0 | 0 | 0 | 100.0 |
| Allocateur de mémoire dynamique O(1) | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 100.0 |
| Ordonnanceur | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 100.0 |
| Préemption | 4 | 0 | 0 | 4 | 0 | 0 | 0 | 100.0 |

Résultats par Plate-forme

| Plate-forme | Total | Non exécuté | [%] | Reussi | [%] | Echec | [%] | Bloqué | [%] | Non Disponible | [%] | Inconnu | [%] | Achévé [%] |
|-------------|-------|-------------|------|--------|-------|-------|-------|--------|-------|----------------|-------|---------|------|------------|
| SODIMM2410 | 25 | 0 | 0.00 | 5 | 20.00 | 8 | 32.00 | 3 | 12.00 | 9 | 36.00 | 0 | 0.00 | 100.00 |