The REBOL Documentation Project

-- FR - Documentation REBOL - Articles Techniques --

Articles Techniques

Le protocole System

Olivier Auverlot

Première publication : 16 mars 2005, et mis en ligne le mercredi 16 mars 2005

Résumé :

Introduit avec la version 2.5.5 de Rebol/Core, le protocole system ne doit pas être confondu avec l'objet system de l'évaluateur. Il s'agit ici d'un accès aux fonctionnalités de bas niveau du système d'exploitation de la machine hôte. Ses deux principaux usages concernent la gestion des signaux sous UNIX et l'interception des événements de Microsoft Windows.

Introduit avec la version 2.5.5 de Rebol/Core, le protocole system ne doit pas être confondu avec l'objet system de l'évaluateur. Il s'agit ici d'un accès aux fonctionnalités de bas niveau du système d'exploitation de la machine hôte. Ses deux principaux usages concernent la gestion des signaux sous UNIX et l'interception des événements de Microsoft Windows.

La gestion des signaux sous Unix

Sous les systèmes de type UNIX, les processus ont la capacité de recevoir des signaux durant leur exécution. Le système ou l'utilisateur peut ainsi donner un ordre à l'application, et celle-ci doit réagir en conséquence. Le plus souvent, un signal est transmis à un processus en utilisant la commande kill suivie du numéro du signal. Avec le protocole system, une application Rebol peut maintenant prendre en compte la réception d'un tel signal. Cela se révèle particulièrement utile si vous écrivez des daemons, c'est-à-dire des applications tournant en permanence sur la machine UNIX. Grâce aux signaux, un processus de ce type peut recevoir l'ordre de stopper son exécution et libérer proprement les ressources utilisées. Le protocole system permet la surveillance de six types de signaux et la mise en place d'un gestionnaire de signaux, dont l'objectif est de réagir à l'arrivée d'un ordre provenant du système.

Signal	Numéro	Signification	Description
SIGHUP	1	Hang Up	Problème sur le terminal ou arrêt d'un processus enfant par le processus parent
SIGINT	2	Interrupt	Le processus a été stoppé en utilisant le clavier du terminal de contrôle par la combinaison de touches CTRL+C.
SIGQUIT	3	Quit	Identique à SIGINT mais sauvegarde un état de la mémoire dans le répertoire courant.
SIGTERM	15	Software Termination	Arrêt du processus
SIGUSR1	16	User Defined Signal 1	Signal définissable par l'utilisateur
SIGUSR2	17	User Defined Signal 2	Signal définissable par l'utilisateur

L'exemple suivant est un script comportant un gestionnaire de signaux, dont l'objectif est d'afficher à l'écran les noms des différents signaux reçus. L'écoute des signaux se fait par l'intermédiaire d'un port créé à l'aide la fonction enable-system-trap. Le script ouvre un nouveau port, puis, à l'aide du mot set-mode, définit la liste des signaux qui seront pris en compte par le gestionnaire. Le paramètre signal-names utilisé par le mot get-modes permet d'initialiser la propriété signal avec l'intégralité des signaux dont dispose l'évaluateur. Une fois le port créé et configuré, celui-ci est ajouté à la liste system/ports/wait-list, qui recense les ports écoutés :

```
enable-system-trap: does [
        if none? attempt [
                system/ports/system: open [ scheme: 'system ]
        ] [
                print "Le port system n'est pas utilisable"
                quit
        ]
        if find get-modes system/ports/system 'system-modes 'signal [
                set-modes system/ports/system [
                        signal: get-modes system/ports/system 'signal-names
                ]
        ]
        append system/ports/wait-list system/ports/system
]
```

Il vous faut maintenant définir une fonction permettant de consulter les signaux détectés par le port d'écoute. La fonction check-system-trap affiche tous les signaux interceptés par le port

system/ports/system :

```
check-system-trap: func [ /local msg ] [
                while [ msg: pick system/ports/system 1 ] [
                        print mold msg
                ]
]
```

Pour rendre utilisable votre script, il vous faut appeler dans votre script la fonction d'initialisation, ainsi que, à intervalle régulier, la fonction check-system-trap :

```
enable-system-trap
forever [check-system-trap ]
```

La figure 1 illustre l'exécution de ce script sur une machine sous Mac OS X. L'envoi de différents signaux à l'aide de la commande kill est bien détecté par le script Rebol.

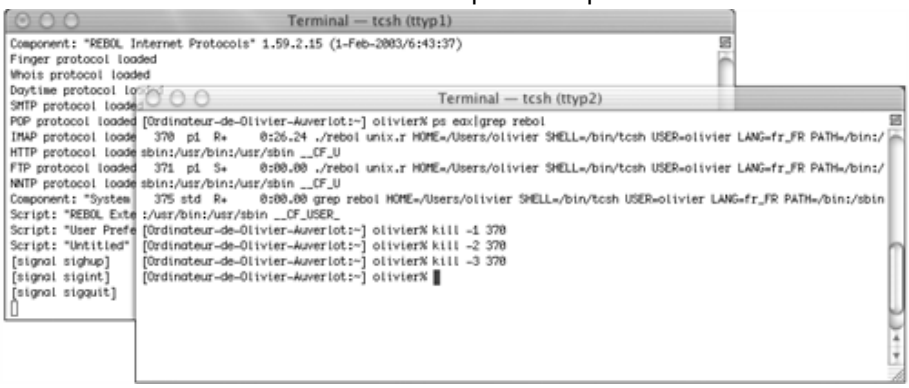

Figure 1 Affichage des signaux interceptés

Intercepter les événements de Windows

Le protocole system est également disponible sous Microsoft Windows. Il s'agit ici d'écouter les différents messages émis par le système d'exploitation et reçus par le script Rebol durant son exécution. Grâce à lui, vous pouvez détecter le mouvement de la souris, la pression d'une touche, le déplacement de la fenêtre de l'évaluateur, etc. Si vous avez des connaissances en programmation Windows, vous pouvez réaliser un nombre incroyable de choses à l'aide du protocole system. L'exemple qui suit est un script capable de détecter le redimensionnement de la fenêtre de l'évaluateur Rebol et l'utilisation de ses menus déroulants. Pour cela, vous devez utiliser respectivement les messages WM_WINSIZE et WM_MENUSELECT :

```
WM_WINSIZE: 5
WM_MENUSELECT: 287
```

La fonction enable-system-trap permet d'initialiser l'usage du protocole system en déclarant un port destiné à l'écoute des messages du système d'exploitation. Si le port est correctement créé, le mot set-modes est utilisé afin de fixer un filtre sur les messages reçus. Les seuls événements devant être pris en compte sont WM_WINSIZE et WM_MENUSELECT :

```
enable-system-trap: does [
        if none? attempt [ system/ports/system: open [ scheme: 'system ] ] [
                print "Il n'y a pas de port system"
                exit
        ]
```

```
        if find get-modes system/ports/system 'system-modes 'winmsg [
                set-modes system/ports/system [
                        winmsg: reduce [
                                WM_WINSIZE
                                WM_MENUSELECT
                        ]
                ]
        ]
        append system/ports/wait-list system/ports/system
]
```

La fonction check-system-trap permet de réagir aux messages reçus par l'application. Pour chacun d'eux, le port system/ports/system fournit un bloc contenant les quatre éléments suivants :

- type de l'information ('winmsg) ;
- numéro du message ;
- wParam et lParam, deux valeurs numériques retournées par le système.

Pour déterminer l'information émise par le système, il vous suffit de consulter le deuxième élément du message. L'exemple ci-dessous indique les nouvelles dimensions du terminal de Rebol/Core lorsque celui-ci est redimensionné et vous informe de l'utilisation des menus déroulants :

```
check-system-trap: func [ /local msg message cons tmp ] [
        message: none
        while [ msg: pick system/ports/system 1 ] [
                message: copy msg
        ]
        if not none? message [
                switch message/2 [
                        5 [
                        cons: open/no-wait/binary [ scheme: 'console ]
                        prin "^(1B)[7n"
                        tmp: next next to-string copy cons
                        close cons
                        tmp: parse tmp ";R"
                        print [
                                to-integer (second tmp) "x"
                                to-integer (first tmp)
                        ]
                        ]
                        287 [
                        print "Vous avez sélectionné un menu déroulant de la console"
                        ]
                ]
        ]
]
```

Pour terminer votre script, il vous faut initialiser le port system. L'écran est effacé, et le programme entre dans une boucle dont la seule action consiste à appeler le gestionnaire de signaux :

```
enable-system-trap
print "^(1B)[J"
forever [ check-system-trap ]
```

Comme illustré à la figure 2, vos actions sur la console de Rebol/Core sont immédiatement détectées par le gestionnaire des messages.

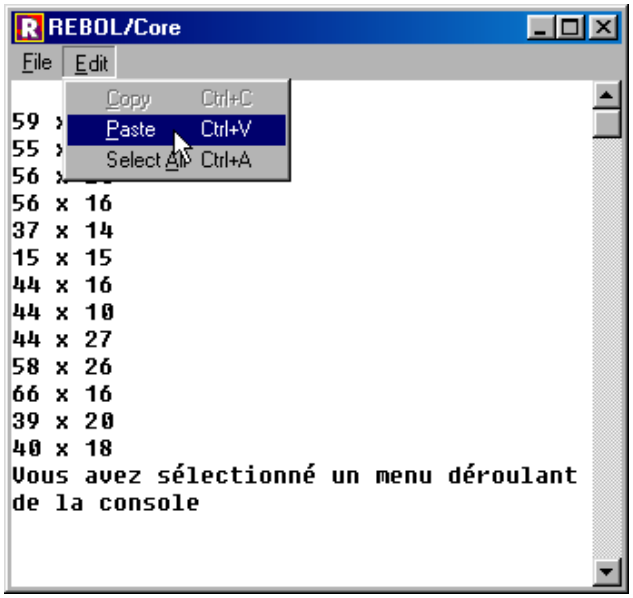

Figure 2. Rebol/Core détecte les actions de l'utilisateur

En résumé, le protocole system permet une parfaite intégration de l'évaluateur Rebol au sein de son environnement d'exécution.

Olivier Auverlot (olivier DOT auverlot AT free DOT fr)