Guide de l'utilisateur de la SIMH, V3.12-3 05-déc-2022

AVIS DE DROIT D'AUTEUR

L'avis de copyright suivant s'applique au code source, au code binaire et à la documentation du SIMH : Code

Introduction

Ce mémorandum présente les simulateurs du SIMH. Ces simulateurs sont des logiciels gratuits ; se référer aux conditions de licence ci-dessus pour les conditions d'utilisation. Aucun support n'est disponible. La meilleure façon de résoudre les problèmes ou d'ajouter des fonctionnalités est de lire et de modifier les sources vous-même. Vous pouvez également envoyer un courrier électronique à simh AT trailing-edge DOT com, mais la réponse n'est pas garantie.

Les simulateurs utilisent une interface de commande commune. Ce mémorandum décrit les caractéristiques de 'interface de commande. Les détails de chaque simulateur sont documentés dans des mémorandums séparés, spécifiques à chaque machine.

1 Compilation et exécution d'un simulateur

Les simulateurs ont été testés sur VAX VMS, Alpha VMS, Alpha UNIX, NetBSD, FreeBSD, OpenBSD, Linux, Solaris, Windows 9x/NT/2000/XP/W7/W10/W11 et MacOS. Le portage vers d'autres environnements nécessitera des modifications du code dépendant du système d'exploitation dans les bibliothèques du SIMH (sim_fio.c, sim_timer.c, sim_console.c, sim_ether.c, sim_sock.c).

Les sources du simulateur sont fournies dans une archive zip et sont organisées de manière hiérarchique. Les fichiers sources des bibliothèques du simulateur se trouvent dans le répertoire de premier niveau ; les fichiers sources de chaque simulateur se trouvent dans des sous-répertoires individuels. Notez que les fichiers inclus dans le répertoire de premier niveau sont référencés à partir des sous-répertoires, sans identifiant de chemin. Votre outil de compilation doit rechercher dans le répertoire de premier niveau les fichiers inclus qui ne sont pas des
dans le répertoire spécifique au simulateur, ou vous devrez copier tous les fichiers des sous-répertoires dans le répertoire principal. Les manifestes de fichiers pour chaque simulateur sont donnés dans la documentation du simulateur.

Les simulateurs reconnaissent ou exigent quelques #définitions à la compilation :

- Les simulateurs 18b exigent que le nom du modèle soit défini dans le cadre de la ligne de commande de compilation (par exemple, PDP4 pour le PDP-4, PDP7 pour le PDP-7, PDP9 pour le PDP-9, PDP15 pour le PDP-15).

- Les simulateurs PDP-10 et IBM 7094 utilisent des variables entières 64b, ce qui nécessite que USE_INT64 soit défini dans la ligne de commande de compilation. Comme les déclarations d'entiers 64b varient, le fichier sim_defs.h contient des déclarations conditionnelles pour Windows (_int64) et Digital UNIX (long). La valeur par défaut est GNU C (long long). Si votre compilateur utilise une convention différente, vous devrez modifier sim_defs.h.

- Les simulateurs PDP-10, PDP-11 et VAX partagent des périphériques communs. Pour distinguer le système cible, l'une des trois variables suivantes doit être définie sur la ligne de commande : VM_PDP10 pour le PDP-10 ; VM_PDP11 pour le PDP-11 ; ou VM_VAX pour le VAX.

- Les simulateurs PDP-11, PDP-11 et VAX supportent optionnellement Ethernet. Pour inclure l'émulation Ethernet, USE_NETWORK doit être défini dans la ligne de commande de compilation. Pour l'instant, le support Ethernet n'a été testé que sur Windows, Linux, NetBSD, OpenBSD, FreeBSD, Solaris, et Alpha VMS, mais il devrait fonctionner dans n'importe quel environnement hôte qui supporte bibliothèque Pcap (voir le fichier readme Ethernet).

- Les simulateurs PDP-11 et VAX prennent optionnellement en charge les disques et les fichiers de périphériques série de plus de 2 Go. Pour inclure le support des gros périphériques, USE_INT64 et USE_ADDR64 doivent être définis dans la ligne de commande de compilation.

Pour démarrer le simulateur, il suffit de taper son nom. (Sous VMS, les simulateurs doivent alors être définis comme des commandes étrangères d'être lancés par leur nom). Le simulateur reconnaît trois options de ligne de commande : -q, -v et -e. Si -q est spécifié, certains messages d'information sont supprimés. Les options -v et -e ne concernent que les fichiers de commande et sont décrites dans la section 3.13.

Le simulateur interprète les arguments de la ligne de commande, s'il y en a, comme le nom du fichier et les arguments d'un `DO` commande :

```
% pdp10 {switches} {<fichier de démarrage> {arg,arg,...}}
```

Si aucun fichier n'est spécifié sur ligne de commande, le simulateur recherche un fichier de démarrage composé du nom du simulateur (y compris ses composants de chemin) et de l'extension .ini. Si un fichier de démarrage est spécifié, soit sur la ligne de commande, soit implicitement via la capacité .ini, il doit contenir une série de commandes non interactives du simulateur, une par ligne. Ces commandes peuvent être utilisées pour définir des paramètres standard, par la taille des disques.

Après avoir initialisé ses structures internes et traité le fichier de démarrage (s'il existe), le simulateur tape son nom et sa version, puis demande une entrée avec :

```
sim>
```

1.1 Compilation sous UNIX/Linux/Mac OS- X

Les sources proviennent d'un système Windows et comportent des cr-lf à la fin chaque ligne. Pour une utilisation sous UNIX ou Mac, les sources doivent être converties aux conventions de texte UNIX ou Mac. Cela peut être fait avec l'utilitaire UNZIP (unzip -a).

Le fichier makefile fourni compilera les simulateurs pour les systèmes UNIX qui supportent le TERMIOS POSIX. Les VAX et PDP-11 peuvent être compilés avec ou sans le support Ethernet. Le fichier makefile compilera automatiquement ces simulateurs avec le support Ethernet si les composants libpcap nécessaires sont disponibles sur le système qui effectue la compilation. Les composants libpcap recommandés sont ceux qui sont empaquetés et fournis par le fournisseur du système d'exploitation hôte. Si le fournisseur de votre système d'exploitation hôte ne distribue pas de composants libpcap (tous les systèmes modernes le font), vous pouvez utiliser la libpcap qui peut être téléchargée et compilée à partir des sources de http://www.tcpdump.org/.

Compilation avec ou sans support Ethernet :

```
gmake {target|ALL|clean}
```

Notes pour la compilation manuelle :

- Le modèle de gestion de terminal UNIX par défaut est l'interface POSIX TERMIOS, qui est supportée par Linux, Mac OS/X, et Alpha UNIX. Si votre UNIX ne supporte que l'interface terminal BSD, BSDTTY doit être défini dans la ligne de commande de compilation.

- Les simulateurs PDP-8, PDP-11, PDP 18b, PDP-10 et Nova utilisent la bibliothèque mathématique. Si votre UNIX ne lie pas automatiquement la bibliothèque mathématique, vous devez ajouter -lm à la ligne de commande de compilation.

Exemples :

- PDP-11 sous TERMIOS UNIX :

  ```
  % cc -DVM_PDP11 pdp11_*.c scp.c sim_*.c -lm -o pdp11
  ```

- PDP-9 sous TERMIOS UNIX :

```
% cc -DPDP9 pdp18b_*.c scp.c sim_*.c -lm -o pdp9
```

- PDP-10 sous BSD terminal UNIX :

```
% cc -DVM_PDP10 -DUSE_INT64 -DBSDTTY pdp10_*.c scp.c sim_*.c -lm -o pdp10
```

1.2 Compilation sous Windows

1.2.1 Compilation avec prise en charge d'Ethernet

Le code Ethernet spécifique à Windows utilise le paquetage WinPCAP 4.x. Ce paquetage pour Windows simule le paquetage libpcap qui est disponible gratuitement pour les systèmes Unix. La construction de simulateurs avec un support Ethernet intégré est possible si les composants WinPcap requis sont disponibles sur votre système de construction au moment de la compilation. Ces composants sont disponibles en téléchargeant le fichier : https://github.com/downloads/simh/simh/windows-build.zip. Ce fichier zip contient un fichier appelé 0ReadMe_Build_Dependencies.txt qui explique comment localiser le contenu du fichier zip décompressé pour construire des simulateurs avec support Ethernet.

Pour qu'un simulateur (construit avec le support Ethernet) fournisse une fonctionnalité Ethernet fonctionnelle, WinPCAP doit être installé sur le système. Pour installer le paquet WinPcap, il faut procéder comme suit :

- Téléchargez la version 4.x à partir de http://www.winpcap.org.
- Installez le paquet comme indiqué.

1.2.2 Compilation sous Visual C++

Visual C++ exige que des projets soient définis pour chaque exécutable en cours de construction. Vous pouvez définir votre propre projet pour chaque simulateur qui vous intéresse ou utiliser les définitions de projet prédéfinies pour cette version de simulateur.

Chaque simulateur doit être organisé comme un projet Visual C++ distinct. En partant d'une application console vide,

- Ajoutez tous les fichiers du manifeste de fichiers du simulateur au projet.
- Ouvrez la boîte de dialogue Propriétés (VC++ .NET).
- Sous C/C++, Category : Général, ajoutez toutes les définitions de préprocesseur requises (par , USE_INT64).
- Sous C/C++, Category : Preprocessor, ajoutez le répertoire de simulation de niveau supérieur aux répertoires d'inclusion supplémentaires. Pour le VAX et le PDP-10, vous devez également ajouter le répertoire PDP-11.
- Sous Link, ajoutez wsock32.lib et winmm.lib à la fin de la liste Object/Module Libraries.
- Si vous construisez le PDP-11 et le VAX avec le support Ethernet, vous devez également ajouter les bibliothèques WinPCAP (packet.lib, wpcap.lib) à la liste des bibliothèques Objet/Module.

Si vous utilisez Visual C++ .NET, vous devez désactiver /Wp64 (avertissement sur les incompatibilités 64b potentielles) et désactiver le traitement Unicode. Vous devrez également désactiver l'avertissement 4996 (fonctions de chaîne "dépréciées") ou abaisser le niveau d'avertissement à /W1. Sinon, les compilations génèreront un grand nombre d'avertissements de conversion erronés.

1.3 Compilation sous OpenVMS

La compilation sur OpenVMS nécessite DEC C. Les simulateurs qui nécessitent 64b (PDP-10 et VAX) ne compileront pas sur OpenVMS/VAX. La distribution du SIMH comprend un fichier de commande MMS descrip.mms compile tous les simulateurs à partir des sources. Exemple de compilation manuelle :

- PDP-8 sous VMS :

```
$ cc scp.c,sim_*.c,[.pdp8]pdp8*.c
$ link/exec=pdp8 scp.obj,sim_*.obj,[.pdp8]pdp8*.obj
```

2 Simulateur Conventions

Un simulateur se compose d'une série de dispositifs, dont le premier est toujours l'unité centrale. Un dispositif se compose de registres nommés et d'une ou plusieurs unités numérotées. Les registres correspondent à l'état du dispositif, les unités aux espaces d'adressage du dispositif. Ainsi, l'unité centrale peut avoir des registres tels que PC, ION, etc. et une unité correspondant à la mémoire principale ; une unité de disque peut avoir des registres tels que BUSY, DONE, etc. et des unités correspondant à des unités de disque individuelles. À l'exception de la mémoire principale, les espaces d'adresses des périphériques sont simulés comme des fichiers non structurés dans le système de fichiers de l'hôte. La commande `SHOW CONFIG` affiche la configuration du simulateur.

Un simulateur conserve le temps termes d'unités arbitraires, généralement une unité de temps par instruction exécutée. Les événements simulés (tels que l'achèvement des E/S) sont programmés à un certain nombre d'unités de temps dans le futur. Le simulateur s'exécute de manière synchrone, en invoquant les processeurs d'événements lorsque les événements simulés sont programmés pour se produire. Même les événements asynchrones, comme les entrées au clavier, sont traités par interrogation à intervalles synchrones. La commande `SHOW QUEUE` affiche la file d'attente des événements du simulateur.

3 Commandes

Les commandes du simulateur se composent d'un verbe de commande, de commutateurs optionnels et d'arguments optionnels. Les commutateurs se présentent sous la forme suivante

```
-<lettre>{<lettre>...}
```

Plusieurs options peuvent être spécifiées séparément ou ensemble : -abcd et -a -b -c -d sont traitées de la même manière. Les verbes, les commutateurs et les autres entrées (à l'exception des noms de fichiers) sont insensibles à la casse.

Toute commande commençant par un point-virgule (;) est considérée comme un commentaire et ignorée.

3.1 Chargement et sauvegarde des programmes

La commande `LOAD` (abréviation `LO`) permet de charger un fichier au format du chargeur binaire :

```
load <filename> {options de mise en œuvre}
```

Les types de formats pris en charge sont spécifiques à l'implémentation. Les options (telles que le chargement dans la plage) sont également spécifiques à l'implémentation.

La commande `DUMP` (abréviation `DU`) permet d'effectuer un dumping de la mémoire au format du chargeur binaire :

```
dump <fichier> {options de mise en œuvre}
```

Les types de formats pris en charge sont spécifiques à l'implémentation. Les options (telles que dump within) sont également spécifiques à l'implémentation.

3.2 Sauvegarde et restauration de l'état de

La commande SAVE (abréviation SA) permet de sauvegarder l'état complet du simulateur dans un fichier. Cela inclut le contenu de la mémoire principale et de tous les registres, ainsi que les connexions d'E/S des périphériques, à l'exception des périphériques réseau (tels que les contrôleurs Ethernet et les multiplexeurs de terminaux) :

```
enregistrer <fichier>
```

La commande RESTORE (abréviation REST, alternativement GET) permet de restaurer un état du simulateur précédemment

sauvegardé : restore <filename>

Note : Le format de fichier SAVE compresse les zéros pour minimiser la taille du fichier.

3.3 Réinitialisation des dispositifs

La commande RESET (abréviation RE) réinitialise un dispositif ou l'ensemble du simulateur à un état prédéfini. Si le commutateur -p est spécifié, le dispositif est réinitialisé à son état de mise sous tension :

```
RÉINITIALISER                      réinitialiser tous les appareils
RESET -p                           met tous les appareils sous tension
RÉINITIALISER TOUS LES APPAREILS   réinitialiser tous les appareils
RESET <périphérique>               réinitialise le dispositif spécifié
```

En règle générale, RESET arrête toute opération d'E/S en cours, efface toute demande d'interruption et ramène l'appareil à un état de repos. Elle n'efface pas la mémoire principale et n'affecte pas les connexions d'E/S.

3.4 Connexion et déconnexion des dispositifs

À l'exception de la mémoire principale et des périphériques réseau, les unités sont simulées sous forme de fichiers non structurés dans le système de fichiers de l'hôte. Avant d'utiliser une unité simulée, l'utilisateur doit spécifier le fichier auquel cette unité doit accéder. La commande ATTACH (abréviation AT) associe une unité et un fichier :

```
ATTACH <unité> <fichier>
```

Si le fichier n'existe pas et que le commutateur -e n'a pas été spécifié, un nouveau fichier est créé et un message approprié est imprimé. Si le commutateur -e a été , un nouveau fichier n'est pas créé et un message d'erreur est imprimé. Si le commutateur -n a été spécifié, un nouveau fichier est créé ou le fichier existant est tronqué à zéro.

Si le commutateur -r est spécifié ou si le fichier est protégé en écriture, ATTACH tente d'ouvrir le fichier en lecture seule. Si le fichier n'existe pas ou si l'unité ne prend pas en charge les opérations en lecture seule, une erreur se produit. Les périphériques d'entrée uniquement, tels que les lecteurs de bandes de papier, et les périphériques dotés d'interrupteurs de verrouillage en écriture, tels que les disques et les bandes, prennent en charge l'opération en lecture seule ; les autres périphériques ne le font pas. Si un fichier est attaché en lecture seule, son contenu peut être examiné mais pas modifié.

Pour les bandes magnétiques simulées, la commande ATTACH peut spécifier le format du fichier image de la bande attachée :

```
ATTACH -f <tape_unit> <format> <fichier>
```

Les formats de fichier d'image de bande actuellement pris en charge sont les suivants :

```
SIMH                               Format du simulateur SIMH
E11                                Format du simulateur E11
```

```
PTC                                 Format PTC
P7B                                 Simulateur Pierce format 7 pistes
```

Le format de la bande peut également être défini à l'aide de la commande `SET` avant

```
ATTACH : SET <tape_unit> FORMAT=<format>.
ATT <unité_de_tape> <fichier>
```

Le format d'un fichier joint peut être affiché à l'aide de la commande `SHOW` :

```
SHOW <tape_unit> FORMAT
```

Pour les émulateurs de terminal basés sur Telnet, la commande `ATTACH` associe l'unité maître à un port TCP/IP :

```
ATTACH <unité> <port>
```

Le port est un nombre décimal compris entre 1 et 65535 qui n'est pas utilisé par les protocoles TCP/IP standard.

Pour les émulateurs Ethernet, la commande `ATTACH` associe l'Ethernet simulé à un périphérique Ethernet physique :

```
ATTACH <unité> <nom du dispositif physique>
```

La commande `DETACH` (abréviation `DET`) rompt l'association entre une unité et un fichier, un port ou un périphérique réseau :

```
DÉTACHER TOUTES LES UNITÉS          détacher toutes les unités
DETACH <unité>                      détachement de l'unité spécifiée
```

La commande `EXIT` effectue un `DETACH ALL automatique`.

3.5 Examiner et modifier State

Quatre commandes permettent d'examiner et de changer d'état :

- `EXAMINE` (abrégé `E`) examine un état
- `DEPOSIT` (abrégé `D`) change d'état
- `IEXAMINE` (interactive examine, abrégé `IE`) examine l'état et permet à l'utilisateur de le modifier de manière interactive.
- `IDEPOSIT` (dépôt interactif, abrégé `ID`) permet à l'utilisateur de changer d'état de manière interactive Les

quatre commandes se présentent sous la forme suivante

```
commande {modificateurs} <liste d'objets>
```

Le dépôt doit également inclure une valeur de dépôt à la fin de la commande.

Il existe quatre types de modificateurs : les commutateurs, le nom de l'appareil ou de l'unité, le spécificateur de recherche et, pour `EXAMINE`, le fichier de sortie. Les commutateurs ont été décrits précédemment. Un nom de dispositif/unité identifie le dispositif et l'unité dont l'espace d'adressage doit être examiné ou modifié. Si aucun périphérique n'est spécifié, l'unité centrale (mémoire principale) est sélectionnée ; si un périphérique mais aucune unité n'est spécifiée, l'unité 0 du périphérique est sélectionnée.

Le spécificateur de recherche fournit des critères pour tester les adresses ou les registres afin de déterminer s'ils doivent être traités. Un spécificateur se compose d'un opérateur logique, d'un opérateur relationnel ou des deux, éventuellement séparés par des espaces.

```
{<opération logique> <valeur>} <opération relationnelle> <valeur>
```

où l'opérateur logique est & (et),| (ou), ou^ (ou exclusif), et l'opérateur relationnel est= ou== (égal),
! ou != (non égal),>= (supérieur ou égal),> (supérieur),<= (inférieur ou égal), ou < (inférieur). Si un opérateur logique est spécifié sans opérateur relationnel, il est ignoré. Si un opérateur relationnel est spécifié sans opérateur logique, aucune opération logique n'est effectuée. Toutes les comparaisons sont non signées.

Le modificateur de fichier de sortie redirige la sortie de la commande vers un fichier au lieu de la console. Un modificateur de fichier de sortie se compose de @ suivi d'un nom de fichier valide.

Les modificateurs peuvent être spécifiés dans n'importe quel ordre. Si plusieurs modificateurs du même type sont spécifiés, les derniers modificateurs ont la priorité sur les premiers. Notez que si le nom du dispositif/de l'unité vient après le spécificateur de recherche, les valeurs de recherche seront interprétées dans le radix de l'unité centrale, plutôt que dans celui du dispositif/de l'unité.

La "liste d'objets" se compose d'un ou de plusieurs des éléments suivants, séparés par des virgules :

`registre`	le registre spécifié
`registre[sub1-sub2]`	les emplacements du tableau de registres spécifiés, à partir de l'emplacement sub1 jusqu'à 'emplacement sub2 inclus
`registre[sub1/longueur]`	les emplacements du tableau de registres spécifiés, à partir de l'emplacement sub1 jusqu'à, mais non compris, sub1+longueur
`register[ALL]`	tous les emplacements du tableau de registres spécifié
`registre1-registre2`	tous les registres commençant par register1 jusqu'au registre2 inclus
`l'adresse`	l'emplacement spécifié
`adresse1-adresse2`	tous les emplacements commençant à l'adresse1 jusqu'à et y compris l'adresse2
`adresse/longueur`	tous les emplacements à partir de l'adresse jusqu'à l'adresse+longueur incluse
`ÉTAT`	tous les registres de l'appareil
`TOUS`	tous les endroits de l'unité

Les commutateurs peuvent être utilisés pour contrôler le format des informations affichées :

`-a`	afficher en ASCII
`-c`	afficher sous forme de chaîne de caractères
`-m`	afficher sous forme de mnémoniques d'instruction
`-o`	affichage en octal
`-d`	affichage sous forme décimale
`-h`	affichage en hexadécimal

Les simulateurs acceptent généralement une entrée symbolique (voir la documentation de chaque

simulateur). Exemples :

`ex 1000-1100`	examiner 1000 à 1100
`de PC 1040`	mettre le PC à 1040
`ie 40-50`	examiner de manière interactive 40:50
`ie >1000 40-50`	examiner interactivement le sous-ensemble de lieux 40:50 qui sont >1000
`ex rx0 50060`	examiner 50060, unité RX 0
`ex rx sbuf[3-6]`	examine SBUF[3] à SBUF[6] en RX

```
de all 0                          mettre la mémoire principale à 0
de &77>0 0                        met à 0 toutes les adresses dont les
                                  bits de  faible ne sont pas nuls
ex -m @memdump.txt 0-7777         vidage de la mémoire dans un fichier
```

Note : pour mettre fin à une commande interactive, il suffit de taper une mauvaise valeur (par exemple, XYZ) lorsqu'une entrée est demandée.

3.6 Évaluation des instructions

La commande `EVAL` évalue une instruction symbolique et renvoie la valeur numérique équivalente. Cette commande est utile pour obtenir des arguments numériques pour une commande de recherche :

```
EVAL <expression>
```

3.7 Exécution d'un programme de simulation

La commande `RUN` (abrégée `RU`) réinitialise tous les dispositifs, dépose son argument (s'il est donné) dans le PC et commence l'exécution. Si aucun argument n'est donné, l'exécution commence au PC actuel.

La commande `GO` *ne* réinitialise *pas* les dispositifs, dépose son argument (s'il est donné) dans le PC et commence l'exécution. Si aucun argument n'est donné, l'exécution commence au PC actuel.

La commande `CONT` (abrégée `CO`) *ne* réinitialise *pas* les dispositifs et reprend l'exécution au PC actuel.

La commande `STEP` (abrégée `S`) reprend l'exécution au PC actuel pour le nombre d'instructions donné par son argument. Si aucun argument n'est fourni, une instruction est exécutée.

La commande `BOOT` (abrégée `BO`) réinitialise tous les appareils et démarre l'appareil et l'unité indiqués dans son argument. Si aucune unité n'est fournie, l'unité 0 est amorcée. L'unité spécifiée doit être attachée.

3.7.1 Contrôle du taux de simulation

Par défaut, le simulateur fonctionne aussi rapidement que possible (bien qu'avec une priorité inférieure à la normale) et consomme toutes les ressources de traitement disponibles sur le système hôte. Cela augmentera la consommation d'énergie (et la température de fonctionnement) de nombreux PC et déchargera la batterie d'un ordinateur portable.

La commande `SET THROTTLE` permet à l'utilisateur de réduire la vitesse d'exécution effective à un nombre déterminé d'instructions par seconde ou à un pourcentage déterminé du temps de calcul total de l'hôte :

```
SET THROTTLE xM                   fixer la vitesse d'exécution à x mips
SET THROTTLE xK                   fixer la vitesse d'exécution à x kips
Régler la vitesse x%              limiter le simulateur à x% du temps de l'hôte
```

L'étranglement n'est disponible que sur les systèmes hôtes qui mettent en œuvre une fonction de retard en temps réel de précision.

La commande `SET NOTHROTTLE` désactive l'étranglement. La commande `SHOW THROTTLE` affiche les paramètres actuels de l'étranglement.

Certains simulateurs mettent en œuvre une forme différente de gestion des ressources, appelée "idling". La marche au ralenti suspend l'exécution simulée chaque fois que le programme exécuté sur le simulateur ne fait rien, et fait tourner le simulateur à plein régime lorsqu'il y a du travail à faire. L'étranglement et la marche au ralenti s'excluent mutuellement.

3.8 Arrêt du simulateur

Les programmes s'exécutent jusqu'à ce que le simulateur détecte une erreur ou une condition d'arrêt, ou jusqu'à ce que l'utilisateur force une condition d'arrêt.

3.8.1 Arrêt détecté par le simulateur Conditions

Ces conditions détectées par le simulateur interrompent la simulation :

- Instruction HALT. Si une instruction HALT est décodée, la simulation s'arrête.
- Point d'arrêt. Le simulateur peut prendre en charge les points d'arrêt (voir ci-dessous).
- Erreur d'E/S. Si une erreur d'E/S se produit pendant la simulation d'une opération d'E/S et que l'indicateur d'arrêt de l'appareil en cas d'erreur d'E/S est activé, la simulation s'arrête généralement.
- Condition du processeur. Certaines conditions du processeur peuvent arrêter la simulation ; elles sont décrites avec différents simulateurs.

3.8.2 Conditions d'arrêt spécifiées par l'utilisateur

La saisie du caractère d'interruption arrête la simulation. Le caractère d'interruption est défini par l'option de console `WRU` (where are you) et est initialement fixé à 005 (^E).

3.8.3 Points d'arrêt

Un simulateur peut offrir des possibilités de points d'arrêt. Il peut définir des points d'arrêt de différents types, identifiés par une lettre (par exemple, E pour exécution, R pour lecture, W pour écriture, etc.) À l'heure , la plupart des simulateurs ne prennent en charge que les points d'arrêt E (exécution).

Un point d'arrêt est associé à un décompte et, éventuellement, à une ou plusieurs actions. Chaque fois que le point d'arrêt est pris, le compte associé est décrémenté. Si le compte est inférieur ou égal 0, le point d'arrêt se produit ; sinon, il est différé. Lorsque le point d'arrêt se produit, les actions optionnelles sont automatiquement exécutées.

Un point d'arrêt est défini par la commande `BREAK` :

```
BREAK {-types} {<plage d'adresses>{[count]},{plage d'adresses...}{;action;action...}
```

Si aucun type n'est spécifié, le type de point d'arrêt par défaut spécifique au simulateur (généralement E pour l'exécution) est utilisé. Si aucune plage d'adresses n'est spécifiée, le PC actuel est utilisé. Comme pour `EXAMINE` et `DEPOSIT`, une plage d'adresses peut être une adresse unique, une plage d'adresses basse-haute, ou une plage relative d'adresses/longueurs. Exemples :

```
PAUSE                           établir une pause E sur le PC actuel
BREAK -e 200                    fixer la pause E à 200
BREAK 2000/2[2]                 set E breaks at 2000,2001 with count = 2 BREAK
100;EX AC;D MQ 0                set E break à 100 avec les actions EX AC et
                                 D MQ 0
RUPTURE 100 ;                   supprimer l'action à la rupture à 100
```

Les points d'arrêt actuellement définis peuvent être affichés à l'aide de la commande `SHOW BREAK` :

```
SHOW {-types} BREAK {ALL|<plage d'adresses>{,<plage d'adresses>...}}
```

Les emplacements comportant des points d'arrêt du type spécifié sont affichés.

Enfin, les points d'arrêt peuvent être effacés par la commande NOBREAK.

3.9 Réglage des paramètres de l'appareil

La commande SET (abrégée SE) modifie l'état d'un ou de plusieurs paramètres de l'appareil :

```
SET <device> <paramètre>{=<valeur},{<paramètre>{=<valeur>},...}
```

ou un ou plusieurs paramètres d'unité :

```
SET <unité> <paramètre>{=<valeur>},{<paramètre>{=<valeur>},...}
```

La plupart des paramètres sont spécifiques au simulateur et à l'appareil. Les lecteurs de disques, par exemple, peuvent généralement être paramétrés comme WRITEENABLED ou write LOCKED ; si un périphérique supporte plusieurs types de lecteurs, la commande SET peut être utilisée pour spécifier le type de lecteur. Si un dispositif (ou une unité) est séquentiel, SET <dev|unit> APPEND positionnera le dispositif ou l'unité à la fin du fichier en cours.

Tous les appareils reconnaissent les paramètres suivants :

```
OCT                                 définit le radix des données= 8
DEC                                 définit le radix des données= 10
HEX                                 définit le radix des données= 16
```

3.10 Affichage des paramètres et de l'état de

La commande SHOW (abrégée SH) permet d'afficher l'état d'un ou de plusieurs paramètres de l'appareil :

```
SHOW {<modificateurs} <périphérique> <paramètre>{=<valeur>},
       {<paramètre>{=<valeur>},...}
```

ou un ou plusieurs paramètres d'unité :

```
SHOW {<modificateurs} <unité> <paramètre>{=<valeur>},
       {<paramètre>{=<valeur>},...}
```

Il existe deux types de modificateurs : les commutateurs et les fichiers de sortie. Les commutateurs ont été décrits précédemment. Le modificateur de fichier de sortie redirige la sortie de la commande vers un fichier au lieu de la console. Un modificateur de fichier de sortie se compose de @ suivi d'un nom de fichier valide.

Tous les appareils utilisent les paramètres RADIX (radix d'affichage), MODIFIERS (liste des modificateurs valides) et NAMES (nom logique). Les autres paramètres de l'appareil et de l'unité sont spécifiques à l'implémentation.

SHOW est également utilisé pour afficher l'état global de la simulation :

```
MONTRER LA CONFIGURATION            montre la configuration du simulateur et
                                    l'état de tous les appareils et unités
MONTRER LES DISPOSITIFS             montre la configuration du simulateur
AFFICHER LES MODIFICATEURS          affiche tous les modificateurs disponibles
AFFICHER LES NOMS                   afficher tous les noms logiques
SHOW QUEUE                          montre la file d'attente des événements du simulateur
AFFICHER L'HEURE                    affiche le temps écoulé depuis le dernier RUN
MONTRER LA VERSION                  affiche la version et les options du simulateur
SHOW <dispositif>                   affiche l'état de l'appareil nommé
```

```
SHOW <unité>                          affiche l'état de l'unité nommée
```

SHOW QUEUE et SHOW TIME affichent le temps en unités spécifiques au simulateur ; en général, une unité de temps représente l'exécution d'une instruction.

3.11 Modification de la configuration du site

Dans la plupart des simulateurs, la commande SET <device> DISABLED supprime le dispositif spécifié de la configuration. Un périphérique DISABLED est invisible pour les programmes en cours d'exécution. Il peut toujours être réinitialisé, mais il ne peut pas être attaché, détaché ou démarré. La commande SET <device> ENABLED rétablit un périphérique désactivé dans une configuration.

La plupart des appareils à plusieurs unités permettent d'activer ou de désactiver des unités :

```
SET <unit> ENABLED SET
<unit> DISABLED
```

Lorsqu'une unité est désactivée, elle n'est pas affichée par SHOW DEVICE.

Les noms de dispositifs standard peuvent être complétés par des noms logiques. Les noms logiques doivent être uniques au sein d'un simulateur (c'est-à-dire qu'ils ne peuvent pas être identiques à un nom d'appareil existant). Pour attribuer un nom logique à un appareil :

```
ASSIGNER <dispositif> <nom du journal>  assigne le nom du journal à l'appareil
```

Pour supprimer un nom logique :

```
DEASSIGN <périphérique>             supprimer le nom logique
```

Pour afficher l'affectation actuelle du nom logique :

```
SHOW <device> NAMES                 afficher le nom logique, s'il y en a un
```

Pour afficher tous les noms logiques :

```
AFFICHER LES NOMS
```

3.12 Console Options

Les options de la console sont contrôlées par la commande SET CONSOLE.

Le terminal de console fonctionne normalement dans la fenêtre de contrôle. En option, le terminal de console peut être connecté à un port Telnet. Cela permet aux systèmes d'émuler un VT100 en utilisant l'émulation de terminal intégrée du client Telnet.

```
SET CONSOLE TELNET=<port> (configuration de la console)      connecter le terminal
console à la session Telnet
                                    sur le port
SET CONSOLE NOTELNET                 désactiver la console Telnet
```

La sortie vers la console peut être enregistrée simultanément dans un fichier :

```
SET CONSOLE LOG=<filename> (enregistrer la sortie de la console
dans un fichier)                    enregistre la sortie de la
console dans un fichier SET CONSOLE NOLOG       désactive la
journalisation
```

La console offre une capacité limitée de remappage des touches :

```
SET CONSOLE WRU=<valeur>          interprète la valeur du code ASCII comme
WRU SET CONSOLE BRK=<valeur>      interprète la valeur du code ASCII comme
BREAK
                                  (0 désactive)
SET CONSOLE DEL=<valeur>          interprète la valeur du code ASCII comme DELETE
SET CONSOLE PCHAR=<valeur>        masque de bits des caractères imprimables dans la
plage
                                  [31,0]
```

Les valeurs sont hexadécimales sur les unités centrales hexagonales, octales sur toutes les autres.

La commande `SHOW CONSOLE` affiche l'état actuel des options de la console :

```
AFFICHER LA CONSOLE               afficher toutes les options de la console
SHOW CONSOLE TELNET               show console Telnet state
SHOW CONSOLE LOG                  show console logging state
SHOW CONSOLE WRU                  montrer la valeur attribuée à WRU
SHOW CONSOLE BRK                  montre la valeur attribuée à BREAK
SHOW CONSOLE DEL                  montre la valeur attribuée à DELETE
SHOW CONSOLE PCHAR                afficher la valeur attribuée à PCHAR
```

`SET CONSOLE` et `SHOW CONSOLE` acceptent tous deux plusieurs paramètres, séparés par des virgules, par exemple,

```
SET CONSOLE WRU=5,DEL=177         définir les valeurs de code pour WRU et DEL
```

3.13 Exécution de la commande Fichiers

Le simulateur peut exécuter des fichiers de commande avec la commande `DO` :

```
DO <fichier> {arguments...} exécute les commandes dans le fichier
```

La commande `DO` permet aux fichiers de commande de contenir des arguments substituables. La chaîne %n, où n est compris entre 1 et 9, est remplacée par l'argument n de la ligne de commande `DO`. La chaîne %0 est remplacée par <fichier>. Les séquences \% et \N sont remplacées par les caractères littéraux % et, respectivement. Les arguments contenant des espaces peuvent être placés entre guillemets simples ou doubles.

Si le commutateur -v est spécifié, les commandes du fichier sont répétées avant d'être exécutées.

Si le commutateur -e est spécifié, le traitement de la commande (y compris les invocations de commandes imbriquées) sera interrompu si une erreur de commande est rencontrée. (Les arrêts de simulation n'interrompent jamais le traitement ; utilisez ASSERT pour détecter les arrêts inattendus). Sans ce commutateur, toutes les erreurs, à l'exception des échecs ASSERT, seront ignorées et le traitement des commandes se poursuivra.

Les commandes DO peuvent être imbriquées jusqu'à dix fois.

Plusieurs commandes sont particulièrement utiles dans les fichiers de commandes. Bien qu'elles puissent être exécutées de manière interactive, elles n'ont qu'une fonctionnalité limitée lorsqu'elles sont utilisées de la sorte.

3.13.1 Affichage d'un texte arbitraire

La commande `ECHO` est un moyen utile d'annoter les fichiers de commandes. `ECHO` affiche son argument sur la console :

```
ECHO <chaîne>                     envoie la chaîne de caractères à la console
```

S'il n'y a pas d'argument, ECHO imprime une ligne blanche sur la console. Ceci peut être utilisé pour fournir un espacement dans l'affichage de la console ou dans le journal.

3.13.2 Simulateur de test État

La commande `ASSERT` teste une condition d'état du simulateur et arrête l'exécution du fichier de commandes si la condition est fausse :

```
ASSERT {<dev>} <reg>{<logical-op><valeur>}<conditional-op><valeur>
```

Si <dev> n'est pas spécifié, CPU est supposé. <reg> est un registre (scalaire ou souscrit) appartenant au périphérique indiqué. Le <conditional-op> et le <logical-op> facultatif sont les mêmes que ceux utilisés pour les "search specifiers" par les commandes `EXAMINE` et `DEPOSIT` (voir ci-dessus). Les <valeurs> sont exprimées dans le radix spécifié pour <reg>, et non dans le radix du dispositif.

Si <oplogique> et <valeur> sont spécifiés, la valeur du registre cible est d'abord modifiée comme indiqué. Le résultat est ensuite comparé à la <valeur> via le <op conditionnel>. Si le résultat est faux, un message "Assertion failed" (échec de l'assertion) est imprimé et tout fichier de commande en cours d'exécution est interrompu. Dans le cas contraire, la commande n'a aucun effet.

Par exemple, un fichier de commandes peut être utilisé pour démarrer un système d'exploitation qui s'arrête après le chargement initial à partir du disque. La commande `ASSERT` est alors utilisée pour confirmer que le chargement s'est bien déroulé en examinant le registre "A" de l'unité centrale pour y trouver la valeur attendue :

```
Fichier de commandes d'amorçage du système d'exploitation
;
ATTACH DS0 os.disk BOOT
DS
Le registre A contient un code d'erreur ; 0= bon démarrage
ASSERT A=0
ATTACH MT0 sys.tape
ATTACH MT1 user.tape RUN
```

Dans l'exemple, si le registre A n'est pas à 0, la commande "ASSERT A=0" sera répercutée, le fichier de commandes sera interrompu avec un message "Assertion failed" (échec de l'assertion). Dans le cas contraire, le fichier de commandes continuera à ouvrir le système d'exploitation.

3.14 Exécution des commandes du système

Le simulateur peut exécuter des commandes du système d'exploitation avec la commande ! (spawn) :

```
! <commande du système d'exploitation de l'hôte>
```

Si aucune commande de système d'exploitation n'est fournie, le simulateur tente de lancer le shell de commande du système d'exploitation hôte.

3.15 Obtenir l'aide de

La commande `HELP` permet d'obtenir des informations sur une commande ou sur toutes les commandes :

```
AIDE                                imprimer tous les messages HELP
HELP <commande>                     imprime l'aide pour la commande
```

3.16 Contrôle Débogage

Certains dispositifs simulés peuvent fournir des impressions de débogage pour aider à diagnostiquer des problèmes complexes. La sortie de débogage peut être envoyée à différents endroits ou être entièrement désactivée :

```
SET CONSOLE DEBUG=STDOUT           sortie de débogage directe vers
stdout SET CONSOLE DEBUG=STDERR    sortie de débogage directe vers
stderr SET CONSOLE DEBUG=LOG       sortie de débogage directe vers le
fichier journal SET CONSOLE DEBUG=<filename> sortie de débogage directe
vers le fichier
SET CONSOLE NODEBUG                désactiver la sortie de débogage
```

Si la sortie de débogage est activée, chaque appareil peut être contrôlé à l'aide de la commande `SET`. Si un appareil n'a qu'un seul drapeau de débogage :

```
SET <device> DEBUG                 active la sortie de débogage du
dispositif SET <device> NODEBUG    désactive la sortie de débogage
de l'appareil
```

Si l'appareil dispose de drapeaux de débogage individuels, nommés :

```
SET <device> DEBUG                 active tous les drapeaux de débogage
SET <device> DEBUG=n1;n2 ;... active les drapeaux de débogage n1, n2,
... SET <device> NODEBUG=n1;n2 ;... désactive les drapeaux de débogage
n1, , ... SET <device> NODEBUG     désactive tous les drapeaux de
débogage
```

Si la sortie de débogage est dirigée vers stdout, elle sera mélangée à la sortie normale du simulateur.

3.17 Quitter le simulateur

`EXIT` (synonymes `QUIT` et `BYE`) redonne le contrôle au système d'exploitation.

Annexe 1 : Dossier Représentations

Toutes les représentations de fichiers sont de type little-endian. Sur les hôtes big-endian, le simulateur effectue automatiquement échanges d'octets nécessaires.

A.1 Disques durs

Les disques durs sont représentés sous forme de fichiers binaires non structurés de 16b pour les simulateurs 12b et 16b, 32b pour les simulateurs 18b, 24b et 32b, et de 64b pour les simulateurs 36b.

A.2 Disquettes Disques

Les disquettes sont représentées sous la forme de fichiers binaires non structurés composés d'éléments de données de 8 bits. Elles sont presque identiques aux images des disquettes du simulateur PDP-8 de Doug Jones, mais il leur manque l'en-tête initial de 256 octets. Un utilitaire de conversion entre les deux formats est facile à écrire.

A.3 Bandes magnétiques

Les bandes magnétiques sont représentées sous la forme de fichiers binaires non structurés composés d'éléments de données de 8 bits. Chaque enregistrement commence par un en-tête de 32 octets, au format little endian. Si l'en-tête de l'enregistrement n'est pas un en-tête spécial, il est suivi de n octets de données de 8 bits, puis d'une répétition de l'en-tête de l'enregistrement de 32 bits. Si le nombre d'octets est impair, l'enregistrement est complété pour obtenir une longueur paire ; l'octet de remplissage est indéfini.

Les bandes magnétiques sont indépendantes de l'endian et cohérentes d'une famille de simulateurs à l'autre. Une bande magnétique produite par le simulateur Nova semblera avoir ses mots de 16b échangés par octet si elle est lue par le simulateur PDP-11.

Les formats étendus et standard du SIMH sont décrits en détail dans le document *SIMH Magtape Representation and Handling*, disponible sur la page Papers on Simulation and Historic Hardware du site SIMH.

La SIMH peut lire et écrire des images de bandes magnétiques au format E11. Le format E11 ne diffère du format SIMH que pour les enregistrements de longueur impaire ; la partie des données des enregistrements E11 n'est pas complétée par un octet supplémentaire.

La SIMH peut lire les images de bandes magnétiques au format TPC. Le format TPC utilise un en-tête d'enregistrement de 16 bits, 0x0000 indiquant la marque du fichier. L'en-tête n'est pas répété à la fin de l'enregistrement. Les enregistrements de longueur impaire sont complétés par un octet supplémentaire.

La SIMH peut lire et écrire des images de bandes magnétiques à sept pistes au format Pierce. Le format Pierce utilise seulement 6 bits de données et un bit de parité dans chaque octet. Le bit de poids fort indique le début de l'enregistrement. La fin du fichier est indiquée par un enregistrement d'un (parfois deux) octets consistant en un code 017 (octal).

A.4 Ligne Imprimantes

La sortie d'une imprimante à lignes est représentée par un fichier ASCII de lignes séparées par le caractère de nouvelle ligne. La surimpression est représentée par une ligne se terminant par un retour plutôt que par une nouvelle ligne.

A.5 Bandes DEC

Les cassettes DEC sont structurées en blocs de longueur fixe. Les cassettes DEC PDP-1/4/7/9/15 utilisent 578 blocs de 256 mots de 32 bits. Chaque mot de 32 bits contient 18 bits (6 lignes) de données. Les cassettes DEC PDP-11 utilisent 578 blocs de 256 mots de 16 bits. Chaque mot de 16b contient 6 lignes de données, avec 2b omises. Ceci est compatible avec les fonctions de vidage des DECtapes PDP-11 et avec le programme PUTR de John Wilson. Les bandes DEC du PDP-8 utilisent 1474 blocs de 129 mots de 16b. Chaque mot de 16b contient 12b (4 lignes) de données. Le PDP-8 OS/8 n'utilise pas le 129ème mot de chaque bloc, et les dumps DECtape OS/8 ne contiennent que 128 mots par bloc. Un utilitaire, DTOS8CVT.C, est fourni pour convertir les dumps DECtape OS/8 au format du simulateur.

Un problème connu dans le format DECtape est que lorsqu'un bloc est enregistré dans un sens et lu dans l'autre, les bits d'un mot sont brouillés (à l'inverse du complément). Le PDP-11 traite ce problème en effectuant une inversion automatique du complément lors des écritures et des lectures inversées. Les autres systèmes laissent ce problème au logiciel.

Le simulateur représente cette différence comme suit. Sur le PDP-11, toutes les données sont représentées sous forme normale. La lecture et l'écriture des données ne sont pas sensibles au sens ; la lecture et l'écriture de toutes les données sont sensibles au sens. Les bandes DEC réelles qui sont lues vers l'avant génèrent des images avec la représentation correcte des données.

Sur les autres systèmes, l'écriture directe crée des données sous forme normale, tandis que l'écriture inverse crée des données sous forme de complément inverse. La lecture directe (et la lecture complète) n'effectue aucune transformation, tandis que la lecture inverse (et la lecture complète) modifie les données en complément inverse. Les bandes DEC réelles qui sont lues en avant
génère des données sous forme normale pour les blocs écrits en avant, et des données complémentaires inversées pour les blocs écrits en arrière, correspondant au format du simulateur.

Annexe 2 : Débogage État

L'état de débogage de chaque unité centrale et appareil simulé est le suivant :

```
légende :            y= exécute le système d'exploitation ou le
                     programme d'exemple d = exécute les diagnostics
                     h= exécute des cas de test générés à la
                     main n = non testé
                     = -sans objet
```

système dispositif	PDP-8	PDP-11	Nova	PDP-1	18b PDP
UNITÉ CENTRALE	y	y	y	y	y
FPU	y	y	-	-	y
SIE/SCI	-	y	-	-	-
console	y	y	y	y	y
ruban adhésif en papier	y	y	y	y	y
lecteur de cartes	-	y	-	-	-
imprimante de ligne	y	y	y	h	y
horloge	y	y	y	-	y
terminal supplémentaire	y	y	y	-	y
disque dur	y	y	y	-	y
disque fixe	y	y	h	-	y
disquette	y	y	y	-	-
tambour	-	-	-	h	h
DECtape	y	y	-	h	y
bande magnétique	y	y	y	-	y

système dispositif	1401	2100	PDP-10	H316	MicroVAX 3900
UNITÉ CENTRALE	y	y	y	h	y
FPU	-	y	y	-	y
SIE/SCI	-	y	y	-	-
console	y	y	y	h	y
ruban adhésif en papier	-	y	n	h	-
lecteur de cartes	y	-	-	-	y
imprimante de ligne	y	y	y	h	y
horloge	-	y	y	h	y
terminal supplémentaire	-	y	y	-	y
disque dur	h	y	y	h	y
disque fixe	-	y	-	h	-
disquette	-	-	-	-	y
tambour	-	y	-	-	-
DECtape	-	-	-	-	-
bande magnétique	y	y	y	h	y

système dispositif	GRI	1620	i16	i32	SDS940
UNITÉ CENTRALE	h	y	d	y	d
FPU	-	y	d	y	-
CEI	-	-	-	-	-
console	h	y	d	y	h
ruban adhésif en papier	h	y	d	y	h
lecteur de cartes	-	y	-	-	-

imprimante de ligne	-	y	d	y	h
horloge	h	-	d	y	n
terminal supplémentaire	-	-	h	y	h
disque dur	-	h	d	y	h
disque fixe	-	-	-	-	h
disquette	-	-	d	d	-
tambour	-	-	-	-	h
DECtape	-	-	-	-	-
bande magnétique	-	-	d	y	h

système dispositif	LGP-30	3000	780
UNITÉ CENTRALE	h	y	y
FPU	-	y	y
CEI	-	y	y
console	h	y	y
ruban adhésif en papier	h	-	-
lecteur de cartes	-	-	y
imprimante de ligne	-	y	y
horloge	-	y	y
terminal supplémentaire	-	y	y
disque dur	-	y	y
disque fixe	-	-	-
disquette	-	-	y
tambour	-	-	-
DECtape	-	-	-
bande magnétique	-	y	y

Historique des révisions (de Rev 2.0 à Rev 3.5)

À partir de la révision 2.7, l'historique détaillé des révisions se trouve dans le fichier

sim_rev.h. Rev 3.5, Sep, 05

- Révision des sources pour une meilleure lisibilité
- Ajouté VAX-11/780

Rev 3.4, mai, 05

- Modèle révisé d'interaction de la mémoire

Rev 3.3, Nov, 04

- Ajout de la prise en charge du PDP-11/VAX DHQ11
- Ajout du support PDP-11/VAX TM02/TM03
- Ajout de la prise en charge de l'émulation spécifique au modèle PDP-11
- Ajout d'un support VAX complet
- Remplacement de SET ONLINE/OFFLINE par SET ENABLED/DISABLED

Rev 3.2, avril 04

- Ajout du simulateur LGP-30/LGP-21
- Ajout d'une capacité de modification globale de SHOW
- Ajout du modificateur global SET DEBUG
- Ajout des modificateurs globaux SHOW DEBUG,RADIX,MODIFIERS,NAME
- Ajout du support de la mémoire physique étendue VAX (Mark Pizzolato)
- Ajout du support VAX RXV21
- Révision de la bibliothèque de multiplexeurs de terminaux pour prendre en charge un nombre variable de lignes par multiplexeur
- Ajout du support du PDP-15 LT19 (1-16 terminaux)

Rev 3.1, déc. 03

- Ajout de la prise en charge des bibliothèques Ethernet Alpha/VMS, FreeBSD, Mac OS/X
- Ajout du support de la virgule flottante et de la minuterie d'intervalle pour Eclipse (de Charles Owen)
- Ajout de la prise en charge de la batterie parallèle du PDP-1
- Ajout de la prise en charge des PDP-8 TSC8-75 et TD8E
- Ajout de la prise en charge des DMA/DMC H316/516, de la bande magnétique et des disques à tête fixe.
- Ajout de la prise en charge de l'historique des instructions PDP-8, PDP-15, 32b Interdata

Rev 3.0, mai 03

- Ajout de la prise en charge des noms logiques
- Ajout de la prise en charge de l'historique des instructions
- Ajout de la prise en charge de plusieurs formats de bande
- Ajout de la prise en charge des adresses 64b
- Ajout de la prise en charge du PDP-4 EAE
- Ajout de la prise en charge du PDP-15 FP15 et XVM

Rev 2.10, Nov, 02

- Ajout de la fonction console Telnet, suppression de l'émulation VT
- Ajout de DO avec des arguments substituables (de Brian Knittel)
- Ajout d'un fichier d'initialisation .ini (de Hans Pufal)
- Ajout du mode silencieux (de Brian Knittel)
- Ajout de la commande ! (de Mark Pizzolato)
- Ajout du support Telnet BREAK (de Mark Pizzolato)
- Ajout de la prise en charge de l'activation/désactivation des appareils
- Ajout de crochets optionnels pour le simulateur pour l'entrée, la sortie, les commandes

- Ajout d'actions de point d'arrêt
- Ajout d'une bibliothèque de simulation de bandes magnétiques
- Ajout d'une horloge programmable PDP-11 KW11P
- Ajout du disque PDP-11 RK611/RK06/RK07
- Ajout d'une bande TMSCP PDP-11/VAX
- Ajout du support Ethernet PDP-11/VAX DELQA (de David Hittner)
- Ajout de la disquette PDP-11/PDP-10 RX211/RX02
- Ajout de la prise en charge de l'autoconfiguration PDP-11/VAX
- Ajout de la prise en charge des vecteurs variables PDP-10/PDP-11/VAX
- Ajouté PDP-1 DECtape
- Ajout de la prise en charge de la batterie série PDP-1, PDP-4 Type 24
- Ajout de la prise en charge du PDP-8 RX28
- Ajout du disque à tête fixe PDP-9 RB09, de l'imprimante à ligne LP09
- Ajout d'une imprimante de ligne HP2100 12845A
- Ajout de la prise en charge de la bande magnétique HP2100 13183
- Ajout de la prise en charge de la ROM de démarrage HP2100
- Ajout de la prise en charge de la liaison interprocesseur HP2100
- Ajouté IBM 1620
- Ajouté SDS 940
- Ajout des systèmes Interdata 16b et 32b
- Ajout de la prise en charge du format de fichier DECtape 16b
- Ajout de la prise en charge des dispositifs à mémoire tampon statique
- Ajout d'un support de fin de support pour les bandes magnétiques
- Ajout de la prise en charge de 50/60Hz pour les horloges à fréquence de ligne
- Ajout du support 7B/8B aux terminaux et multiplexeurs
- Ajout du support BREAK aux terminaux et multiplexeurs

Rév. 2.9, janv. 02

- Ajout de tableaux de registres circulaires
- Remplacement de ENABLE/DISABLE par SET ENABLED/DISABLED
- Remplacement de LOG/NOLOG par SET LOG/NOLOG
- Généralisation du paquet d'étalonnage de la minuterie
- Ajout de routines supplémentaires à la bibliothèque des multiplexeurs
- Ajout des commandes SET DISCONNECT et SHOW STATISTICS aux multiplexeurs.
- Réimplémentation du PDP-8 TTX en tant que multiplexeur unifié
- Mise en place d'une file d'attente PC dans la plupart des simulateurs
- Ajout d'un simulateur VAX
- Ajout du simulateur GRI-909
- Ajout du simulateur MITS 8080/Z80 de Peter Schorn
- Ajouté le simulateur IBM 1130 de Brian Knittel
- Ajout de dispositifs HP2100 DQ, DR, MS, MUX
- Ajout des commandes SET VT/NOVT

Rev 2.8, déc. 01

- Ajout de la commande DO
- Ajout d'une fonction de point d'arrêt général
- Ajout d'une capacité SET/SHOW étendue
- Remplacement de ADD/REMOVE par SET ONLINE/OFFLINE
- Reconnaissance du nom du registre mondial
- Ajout de tableaux de registres basés sur l'unité
- Ajout du simulateur System 3 de Charles Owen
- Ajout de la carte du bus d'E/S du PDP-11
- Ajouté PDP-11/VAX RQDX3
- Ajouté PDP-8 RL8A
- Révisé 18b Structure d'interruption PDP
- Révision de la structure des répertoires et de la documentation

- Ajout de la prise en charge de l'environnement MINGW

Rev 2.7, Sep, 01
- Ajout de DZ11 (de Thord Nilson et Art Krewat) aux PDP-11, PDP-10
- Ajout de terminaux supplémentaires au PDP-8
- Ajout du format de caractères emballés TSS/8 au PDP-8
- Ajout des bibliothèques sim_sock et sim_tmxr
- Ajout des fonctions sim_qcount et simulator exit detach all
- Ajout du support de sim_sock pour Macintosh (de Peter Schorn)
- Ajout du niveau de révision du simulateur, de la version SHOW
- Changement de int64/uint64 en t_int64/t_uint64 pour Windows
- Correction d'un bogue dans l'acquittement des interruptions du PDP-11
- Correction de bogues dans la vérification du PDP-11 TS NXM, le code de démarrage, l'état d'erreur ; ajout de caractéristiques et d'états étendus.
- Correction d'un bug dans les fonctions PDP-11 TC stop, stop all
- Correction du bogue de l'interruption de réception en cas de déconnexion dans le DZ11
- Correction des bogues liés aux opérations multi-unités et des bogues liés aux interruptions dans le PDP-11
- RP, PDP-10 RP, PDP-10 TU
- Correction du bogue de détection de la porteuse dans les PDP-11, PDP-10 DZ
- Correction d'un bogue dans la routine de réinitialisation du PDP-8
- Correction d'une conditionnalité dans l'unité centrale PDP-18b
- Correction du bogue SC= 0 dans le PDP-18b EAE
- Correction d'un bogue dans le PDP-7 LPT
- Mise à jour du second terminal Nova pour utiliser sim_tmxr
- Mise à jour du deuxième terminal du PDP-18b pour utiliser sim_tmxr
- Mise à niveau du PDP-11 LTC vers le KW11-L complet
- Suppression de la prise en charge des consoles multiples de piratage

Rev 2.6b, août 01
- Ajout d'un simulateur H316/516
- Ajout d'un support Macintosh par Louis Chrétien, Peter Schorn et Ben Supnik
- Ajout de l'option "bad block table" au PDP-11 RL, RP
- Suppression du registre dans les déclarations
- Correction de bugs trouvés par Peter Schorn
    - erreur endian dans PDP-10, PDP-11 RP
    - erreur d'inversion de l'espace dans le PDP-11 TS
    - saisie symbolique en 1401
- Correction d'un bug dans le chargeur RIM du PDP-1 trouvé par Derek Peschel
- Correction d'un bug dans le disque à tête fixe de Nova

Rev 2.6a, juin 01
- Ajout de l'option API PDP-9, PDP-15
- Ajout d'un deuxième terminal pour le PDP-9 et le PDP-15
- Ajout de l'option PDP-10 pour TOPS-20 Correction d'un bug de la V4.1
- Ajout de l'option CTRL-C du PDP-10 FE pour Windows
- Ajout de la journalisation de la console
- Ajout de la prise en charge de plusieurs consoles
- Ajout de la reconnaissance des commentaires
- Augmentation de la taille des tampons de chaîne pour les noms de chemin longs
- Correction d'un bug dans les E/S big-endian trouvé par Dave Conroy
- Correction de la réinitialisation de DECtape dans PDP-8, PDP-11, PDP-9/15
- Correction de la gestion du PC du chargeur RIM dans le PDP-9/15
- Correction des pointeurs indirects dans la pagination du PDP-10
- Correction de la gestion du SSC dans le PDP-10 TM02/TU45
- Correction du JMS vers une mémoire inexistante dans le PDP-8
- Correction de la gestion des erreurs dans le fichier de commande

Rev 2.6, mai, 01

- Ajout de dispositifs d'activation/désactivation
- Ajouté SHOW DEVICES
- Ajout de l'examen/modification des tableaux de registres
- Ajout d'un simulateur PDP-10
- Ajout de l'autocalibrage de l'horloge pour SCP, Nova, PDP-8, PDP-11, PDP-18b
- Ajouté PDP-8, PDP-11, PDP-9/15 DECtape
- Ajouté PDP-8 DF32
- Ajout du démarrage du moniteur de disque 4k pour les PDP-8 RF08 et DF32
- Ajout de la prise en charge du chargeur de format amusant PDP-4/7
- Ajout de la gestion des extensions aux chargeurs PDP-8 et -9/15
- Ajouté PDP-11 TS11/TSV05
- Ajout d'une minuterie à intervalles entiers pour SCP
- Ajout de l'argument "nom de fichier" à LOAD/DUMP
- Révision des bandes magnétiques et des bootstraps DECtape pour qu'ils se rembobinent avant la première instruction
- Correction de la séquence de rupture de données de 3 cycles dans le PDP-8 RF
- Correction de la séquence de rupture de données de 3 cycles dans le PDP 18b , MT, RF
- Correction de l'écriture CS1.TRE, de l'écriture CS2.MXF,UPE, et de l'écriture CS2.UAI dans PDP-11 RP
- Correction de la définition de la taille de la mémoire de 4M dans le PDP-11
- Correction d'un bogue d'attachement dans RESTORE
- Correction du bug de détachement pour les appareils en mémoire tampon
- Mise à jour des mentions de copyright, correction des commentaires

Rev 2.5a, déc, 00

- Ajout d'un flop CMD pour les imprimantes à bande de papier et les imprimantes de ligne HP
- Ajout d'une entrée d'état pour le perforateur de bande de papier HP et le TTY
- Ajout de la routine de démarrage de la bande magnétique 1401 de Charles Owen
- Ajout du traceur Nova de Bruce Ray et des modules du second terminal
- Ajout de la prise en charge de l'unité centrale Eclipse de Charles Owen
- Ajout de la prise en charge du chargeur RIM/BIN du PDP-9/PDP-15
- Ajout des registres d'état initial de l'extension et de la banque du PDP-9/PDP-15
- Ajout de la prise en charge du duplex half/full par le PDP-9/PDP-15
- La documentation du logiciel a été déplacée dans un fichier séparé
- Correction de la gestion SCP des appareils sans unités
- Correction de l'initialisation FLG, FBF dans de nombreux périphériques HP
- Correction de 1401 bogues trouvés par Charles Owen
    - Les NOPs de 4 et 7 caractères sont légaux
    - 1 char B est enchaîné BCE
    - MCE déplace le caractère entier, et non le chiffre, après le premier.
- Correction des bogues Nova trouvés par Bruce Ray
    - pièges implémentés sur le Nova 3 ainsi que sur le Nova 4
    - DIV et DIVS 0/0 : report de valeur
    - RETN définit la PS à partir de la PF dès le départ
    - IORST n'efface pas le report
    - Nova 4 implémente deux instructions non documentées
- Correction de bogues dans les PDP 18b
    - Calcul de l'adresse indirecte XCT
    - instructions d'indexation manquantes dans le PDP-15
    - traitement du mode banque dans le PDP-15

Rev 2.5, Nov, 00

- Suppression de Digital et Compaq des droits d'auteur, avec l'autorisation de Bill Strecker, vice-président principal de Compaq.
- Révision du format de sauvegarde/restauration pour les simulateurs 64b
- Ajout d'un examen au fichier

- Ajout de types de données entières non signées dans sim_defs
- Ajout des instructions Nova 3 et Nova 4 à Nova CPU
- Ajouté HP2100
- Correction d'une boucle indirecte à travers autoinc/dec dans Nova CPU
- Correction du test MDV activé dans Nova CPU

Rev 2.4, janvier 99
- Placement de toutes les sources sous une licence open source de type X11
- Ajout de la commande DUMP, révision de l'interface sim_load
- Ajout de la commande SHOW MODIFIERS
- Révision du format de la bande magnétique pour inclure un indicateur d'erreur d'enregistrement
- Correction des problèmes de 64b dans SCP
- Correction d'un problème de big endian dans la routine de mauvais bloc du PDP-11
- Correction d'un bogue d'interruption en cas d'erreur dans les disques PDP-11 RP/RM
- Correction de l'inversion ROL/ROR dans les routines symboliques du PDP-11

Rev 2.3d, Sep, 98
- Ajout du support BeOS
- Ajout de commandes et de commutateurs radix
- Ajout de la prise en charge du CIS PDP-11
- Ajout de RT11 V5.3 aux kits de distribution
- Correction des bogues "shift 32" dans SCP, PDP-11 en virgule flottante
- Correction d'un bogue dans le lecteur de bande papier du PDP-11
- Correction d'un bug dans la gestion de ^D

Rev 2.3c, mai 98
- Correction d'un bogue dans la vérification du dépassement de capacité du PDP-11 DIV
- Correction de bogues dans l'amorçage de la bande magnétique du PDP-11
- Correction d'un bogue dans la sélection de l'unité de bande magnétique du PDP-11
- Remplacement des images de disques UNIX V7

Rev 2.3b, mai, 98
- Ajout de la reconnaissance des commutateurs à toutes les commandes du simulateur
- Ajout d'un chargeur RIM au lecteur et chargeur de bandes de papier du PDP-8
- Ajout d'un deuxième bloc d'amorçage à la bande magnétique du PDP-11
- Correction d'un bug dans l'amorçage du PDP-8 RF
- Correction d'un bug dans l'affichage symbolique du PDP-11
- Correction de bogues dans la virgule flottante du PDP-11 (LDEXP, STEXP, MODf, STCfi, gestion des débordements).

Rev 2.3a, Nov, 97
- Ajout d'une fonction de recherche
- Ajout de la commande "bad block table" pour les disques PDP-11
- Ajout d'un bootstrap sur la bande magnétique du PDP-11
- Ajout de disques à tête mobile Nova supplémentaires
- Ajout du logiciel d'échantillonnage RT-11
- Correction de bogues dans les disques PDP-11 RM/RP
- Correction de bogues dans les disques à tête mobile Nova
- Correction de la dépendance endienne dans le chargeur RIM du PDP 18b

Rev 2.3, mars 97
- Ajouté PDP-11 RP
- Ajout du PDP-1
- Changement des E/S de terminal UNIX en TERMIOS
- Changement du format de la bande magnétique pour une bande à double extrémité
- Changement du mnémonique de la page courante du PDP-8 de T à C
- Ajout de routines d'E/S indépendantes de l'endian

- Ajout de types de données entières précises
- Correction d'un bug dans sim_poll_kbd
- Correction d'un bogue dans le chargeur binaire du PDP-8
- Correction de bogues dans la bande magnétique TM11
- Correction d'un bug dans le bootstrap de RX11
- Correction d'un bogue dans 18b PDP ADD
- Correction d'un bogue dans le lecteur de bande papier PDP 18b
- Correction d'un bug dans la console PDP-4
- Correction d'un bogue dans l'imprimante PDP-4, 7 lignes

Rev 2.2d, déc, 96
- Ajout des commandes ADD/REMOVE
- Ajout de la prise en charge de l'activation/désactivation de l'unité dans les simulateurs d'appareils
- Fonctionnalités ajoutées pour le projet IBM 1401
- Ajout de la reconnaissance des commutateurs pour la saisie symbolique
- Correction d'un bug dans la longueur variable de l'IEXAMINE
- Correction du bug de l'écran LCD dans le RX8E
- Changements initiaux pour Win32
- Ajouté IBM 1401

Rev 2.2b, avril, 96
- Ajout de la prise en charge de la taille de la mémoire dynamique du PDP-11

Rev 2.2a, févr. 96
- Nouveau format de bande magnétique indépendant de l'endian

Rev 2.2 Jan, 96
- Ajout de tampons de registre pour la sauvegarde/restauration
- Ajout de 18b PDP
- Garantie que les temps TTI et CLK ne sont pas nuls
- Correction d'un bug dans l'interaction entre le point de rupture et la course à pied
- Correction d'un bogue concernant le retour en arrière de la bande magnétique jusqu'à EOF
- Correction de l'inversion ISZ/DCA dans la table des symboles du PDP-8
- Correction de la conversion à six bits dans l'examen/dépôt PDP-8
- Correction du bogue d'incrémentation de l'origine dans le chargeur binaire du PDP-11
- Correction d'un bogue d'optimisation de GCC longjmp dans le processeur PDP-11
- Correction d'un bug dans le calcul du nombre d'unités dans SCP et dans les disques à tête mobile Nova, PDP-11, 18b PDP.

Rev 2.1 Déc, 95
- Correction d'un bug PTR (réglage effectué sur EOF) dans PDP-8, Nova
- Correction du bogue RX (erreur de réglage lors de l'INIT si le lecteur 1 n'est pas connecté) dans PDP-8, PDP-11
- Correction du traitement RF du drapeau de la cellule photoélectrique dans le PDP-8
- Correction d'un bug de taille automatique (choix du plus petit disque en cas de nouveau fichier) dans PDP-11, Nova
- Correction d'un bogue de non-attachement (signalé comme n'étant pas attachable) dans la plupart des dispositifs de stockage de masse
- Correction des ROM de démarrage de Nova
- Correction d'un bug dans RESTORE (la demande n'était pas faite si le délai= 0)
- Correction d'un bogue dans RESTORE (position de l'appareil bloquée)
- Les tableaux statiques constants sont déclarés comme statiques const.
- Ajout de simulateurs de bandes magnétiques PDP-8, Nova
- Ajout du mode Dasher au simulateur de terminal Nova
- Ajout du support LINUX

Rev 2.0 Mai, 95
- Ajout de l'assemblage/désassemblage symbolique

Remerciements

Le SIMH n'aurait pas été possible sans 'aide de personnes du monde entier. Je tiens à remercier personnes suivantes, qui ont toutes consacré leur temps et leur talent à ce projet d'"archéologie informatique" :

Bill Ackerman	Consultation sur le PDP-1
Alan Bawden	Conseil ITS Winfried
Bergmann	Tests de portage
Linux Phil Budne	Tests de portage
Solaris	
Max Burnet	Informations, documentation et logiciels PDP Robert
Alan Byer	Support et test des sockets VMS
James Carpenter	Tests de portage sous LINUX
Chip Charlot	PDP-11 RT-11, RSTS/E, RSX-11M autorisations légales
Louis Chrétien	Portage sur Macintosh
Dave Conroy	Documentation HP 21xx, débogage PDP-10, PDP-18b L Peter
Deutsch	Logiciel LISP pour le PDP-1
Ethan Dicks	Débogage du PDP-11 2.9 BSD
John Dundas	Débogage de l'unité centrale du PDP-11, simulateur d'horloge
programmable Jonathan Engdahl	Débogage de dispositifs PDP-11
Carl Friend	Documentation Nova et Interdata, et logiciel RDOS Megan
Gentry	Débogage des entiers du PDP-11
Dave Gesswein	Documentation du PDP-8 et du PDP-9.15, images du PDP-8 sur DECtape, disque et bande papier, images du PDP-9/15 sur DECtape
Dick Greeley	Permissions légales pour le PDP-8 OS/8 et le PDP-10 TOPS-
10/20 Gordon Greene	Source lisible en machine du PDP-1 LISP
Lynne Grettum	PDP-11 RT-11, RSTS/E, RSX-11M autorisations légales
Franc Grootjen	Débogage du PDP-11 2.11 BSD
Doug Gwyn	Débogage de la portabilité
Kevin Handy	Documentation TS11/TSV05, fichier make
Ken Harrenstein	Simulateur KLH PDP-10
Bill Haygood	Informations sur le PDP-8, simulateur et logiciel
Wolfgang Helbig	Implémentation du DZ11
Mark Hittinger	Débogage du PDP-10
Dave Hittner	Débogage SCP, émulateur DEQNA et bibliothèque Ethernet
Sellam Ismail	Documentation GRI-909
Jay Jaeger	Conseil IBM 1401
Doug Jones	Informations sur le PDP-8, simulateur et logiciel
Brian Knittel	Simulateur IBM 1130, extensions SCP pour le support de l'interface graphique
Al Kossow	Documentation et logiciels HP 21xx, Varian 620, TI 990, Interdata, DEC Arthur
Krewat	Changements DZ11 pour le PDP-10
Mirian Crzig Lennox	Débogage de l'ITS et du DZ11
Don Lewine	Documentation Nova et autorisations légales
Tim Litt	Documentation et schémas du matériel du PDP-10, images de bandes et sources de logiciels
Tim Markson	Débogage du DZ11
Bill McDermith	Débogage HP 2100, simulateur 12565A
Scott McGregor	PDP-11 UNIX autorisations légales
Jeff Moffatt	Informations, documentation et logiciels HP 2100 Alec
Muffett	Test des ports Solaris
Terry Newton	Débogage de la HP 21MX
Thord Nilson	Mise en œuvre du DZ11
Charles Owen	Débogage du disque à tête mobile Nova, simulateur Altair, simulateur Eclipse, Simulateur IBM System 3, diagnostic IBM 1401, débogage et démarrage sur bande magnétique

Sergio Pedraja Débogage de l'environnement
MINGW Derek Peschel Débogage du PDP-10
Paul Pierce Diagnostic IBM 1401, récupération des supports
Mark Pizzolato Améliorations des simulateurs SCP, Ethernet et VAX
Hans Pufal Débogage du PDP-10, démarrage du PDP-15, récupération du DOS-15, documentation du DOS-15, restauration du PDP-9
Bruce Ray Logiciels, documentation, corrections de bogues et nouveaux périphériques pour le Nova, portage OS/2 Craig St Clair Documentation DEC
Richard Schedler Maintenance des dépôts publics
Peter Schorn Portage sur Macintosh
Stephen Schultz Débogage du PDP-11 2.11 BSD
Olaf Seibert Test du portage NetBSD
Brian & Barry Silverman Simulateur et logiciel PDP-1
Tim Shoppa Documentation Nova, logiciels RDOS, archives des logiciels PDP-10 et PDP-11, hébergement du site SIMH
Van Snyder IBM 1401 zero footprint bootstraps
Michael Somos Débogage du PDP-1
Hans-Michael Stahl Tests de portage OS/2, mise en œuvre de
TERMIOS Tim Stark Simulateur TS10 PDP-10
Larry Stewart Suggestion initiale pour le projet Bill
Strecker Permission de renverser les droits
d'auteur Chris Suddick Débogage du PDP-11 en virgule
flottante Ben Supnik Routine de synchronisation Macintosh
Bob Supnik Simulateurs SIMH
Ben Thomas Routines d'E/S caractère par caractère du VMS
Warren Toomey Logiciel UNIX pour PDP-11
Deb Toivonen Documentation DEC
Mike Umbricht Documentation DEC, documentation et schémas H316 Leendert
Van Doorn Débogage du PDP-11 UNIX V6, implémentation de TERMIOS
Fred Van Kempen Code Ethernet, émulateur RK611, débogage PDP-11, débogage VAX/Ultrix Holger Veit Support des sockets OS/2
David Waks Logiciel PDP-8 ESI-X et PDP-7 SIM8 Tom
West Documentation Nova
Adrian Wise Simulateur H316, documentation et logiciel John
Wilson Simulateur et logiciel PDP-11
Joe Young Débogage RP sur Ultrix-11 et BSD

En , les sociétés suivantes ont gracieusement mis leurs logiciels à disposition des amateurs : Data General Corporation
Digital Equipment Corporation Compaq
Computer Corporation Mentec
Corporation
La Corporation de l'opération Caldera
de Santa Cruz
Hewlett-Packard Corporation