Haiku, un système d'exploitation libre

PulkoMandy / Adrien Destugues

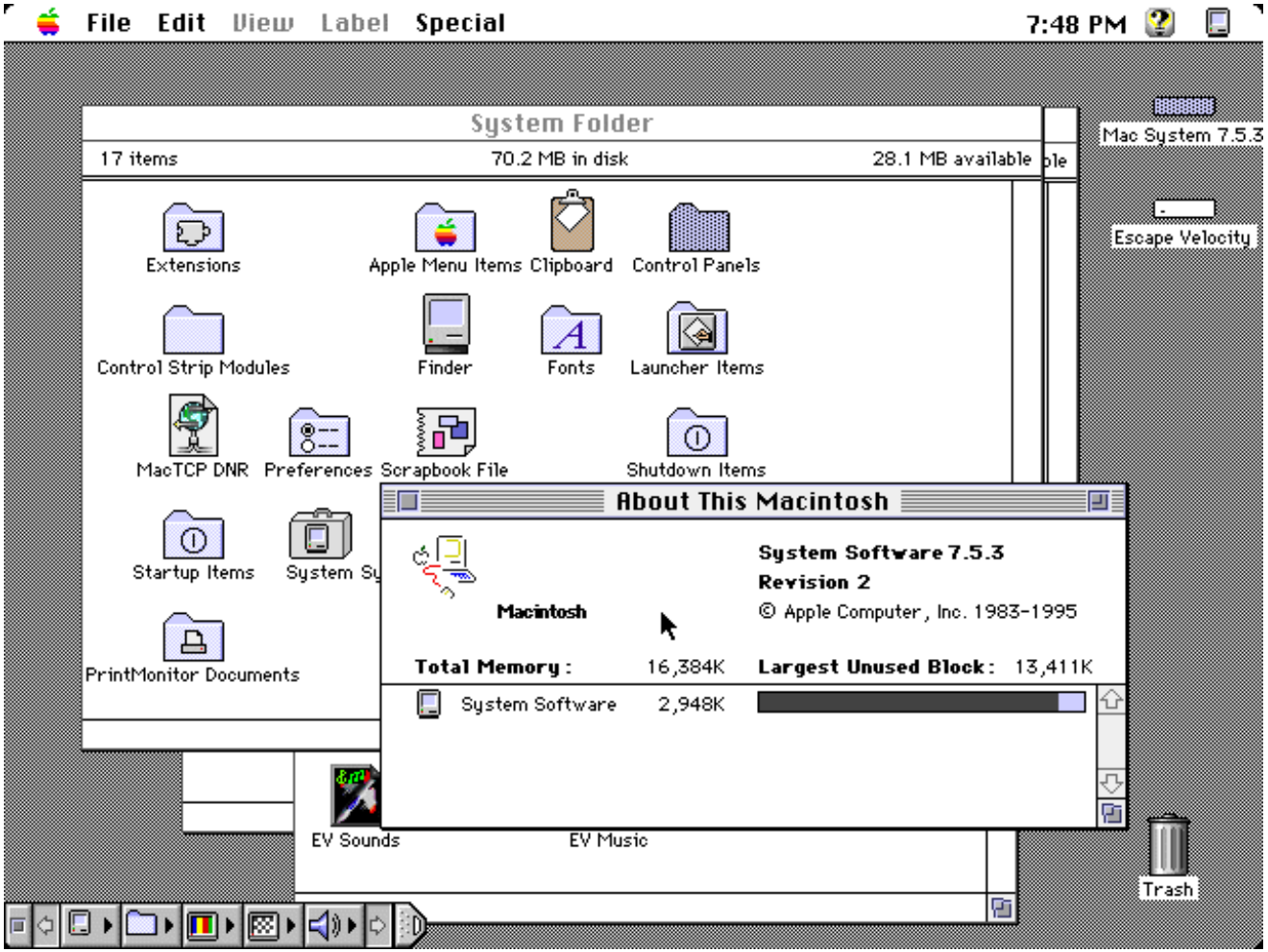
VIVERIS

Qu’est-ce que Haiku ?

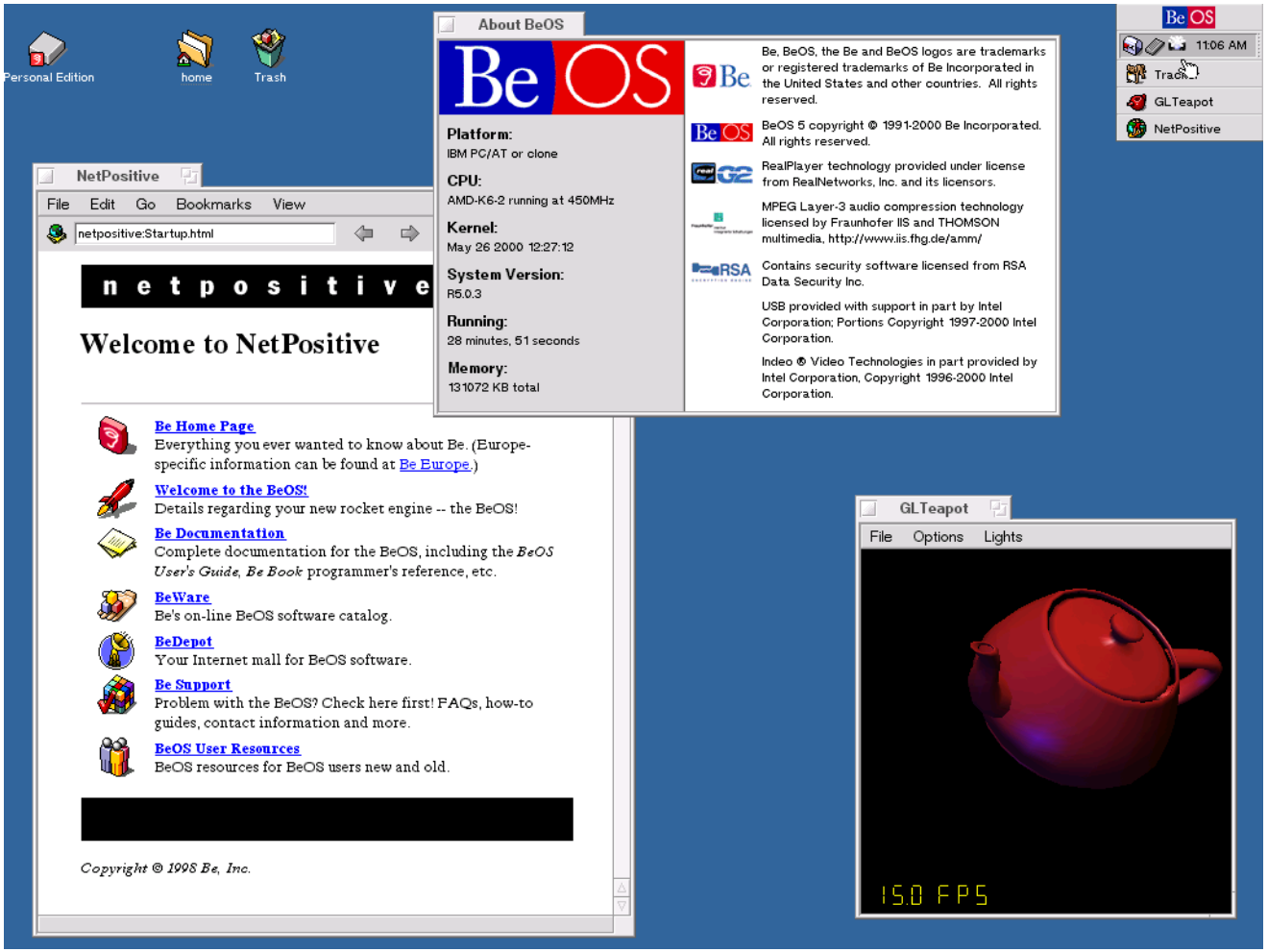
- Système d’exploitation pour les ordinateurs personnels
- Logiciel libre
- Facile à utiliser, mais approprié pour les utilisateurs « avancés »

Historique

Historique


File Edit View Label Special
7:48 PM
System Folder
17 items
70.2 MB in disk
28.1 MB available
Extensions
Apple Menu Items
Clipboard
Control Panels
Control Strip Modules
Finder
Fonts
Launcher Items
MacTCP DNR
Preferences
Scrapbook File
Shutdown Items
Startup Items
PrintMonitor Documents
About This Macintosh
Macintosh
System Software 7.5.3
Revision 2
© Apple Computer, Inc. 1983-1995
Total Memory: 16,384K
Largest Unused Block: 13,411K
System Software 2,948K
EV Sounds
EV Music
Mac System 7.5.3
Escape Velocity
Trash

Historique


Personal Edition
home
Trash
About BeOS
BeOS
Platform:
IBM PC/AT or clone
CPU:
AMD-K6-2 running at 450MHz
Kernel:
May 26 2000 12:27:12
System Version:
R5.0.3
Running:
28 minutes, 51 seconds
Memory:
131072 KB total
Be, BeOS, the Be and BeOS logos are trademarks or registered trademarks of Be Incorporated in the United States and other countries. All rights reserved.
BeOS 5 copyright © 1991-2000 Be Incorporated. All rights reserved.
RealPlayer technology provided under license from RealNetworks, Inc. and its licensors.
MPEG Layer-3 audio compression technology licensed by Fraunhofer IIS and THOMSON multimedia, http://www.iis.fhg.de/amm/
Contains security software licensed from RSA Data Security Inc.
USB provided with support in part by Intel Corporation; Portions Copyright 1997-2000 Intel Corporation.
Indeo ® Video Technologies in part provided by Intel Corporation, Copyright 1996-2000 Intel Corporation.
BeOS
1:06 AM
Tracker
GLTeapot
NetPositive
NetPositive
File Edit Go Bookmarks View
netpositive:Startup.html
netpositive
Welcome to NetPositive
Be Home Page
Everything you ever wanted to know about Be. (Europe-specific information can be found at Be Europe.)
Welcome to the BeOS!
Details regarding your new rocket engine -- the BeOS!
Be Documentation
Complete documentation for the BeOS, including the BeOS User's Guide, Be Book programmer's reference, etc.
BeWare
Be's on-line BeOS software catalog.
BeDepot
Your Internet mall for BeOS software.
Be Support
Problem with the BeOS? Check here first! FAQs, how-to guides, contact information and more.
BeOS User Resources
BeOS resources for BeOS users new and old.
Copyright © 1998 Be, Inc.
GLTeapot
File Options Lights
15.0 FPS

Pourquoi Haiku ?

Technodiversité et innovation

Un vrai OS libre pour les ordinateurs personnels

Défi technique et humain

Nostalgie ?

Un logiciel libre

Liberté :

- D’utiliser le programme
- De l’étudier et de le modifier
- De le distribuer
- De distribuer les versions modifiées

La licence MIT

Licence avec peu de restrictions

Compatible avec du code non libre

Choix pragmatique plutôt que militant

Histoire de Haiku

2001 : création du projet

2002 : la fenêtre de configuration de l’écran de veille

2004 : OpenBeOS devient Haiku

2005 : Haiku démarre « jusqu’au bureau »

2008 : Haiku peut se compiler lui-même

2009 : version Alpha 1

2018 : version Beta 1

Organisation

En chiffres

5 millions de lignes de code

1400 commits par an (3 à 4 par jour)

60 contributeurs (parmi les 2 % plus gros projets)

- Mais à peine une dizaine vraiment actifs
- Dont un qui a fait 43 % des commits en 2019

Traduit dans 23 langues

Plusieurs centaines d’utilisateurs

Un projet communautaire

Contributeurs participant sur leur temps libre

+ contrats, GSoC, Outreachy

Tout le monde peut participer

Programmation, traduction, documentation, graphisme, UX design, ...

Équipe répartie partout autour du monde

Organisation

Pas de chef !

Décisions prises par consensus

Par l'équipe du projet (contributeurs recrutés par cooptation)

On choisit les personnes qui comprennent « the Haiku way »

Une vision commune

Éviter les « forks », plutôt intégrer les bonnes idées

Cependant, garder une ligne directrice cohérente

« The Haiku Way »

Réglages par défauts sensés, plutôt que tout rendre configurable

- Prise en main plus facile
- Contrôle qualité plus simple
- Documentation plus claire

Supposer le probable, permettre l'inhabituel

- Autrement dit, « ça juste marche »

Bonnes pratiques pour l'interface utilisateur

- Unification entre les applications
- Éviter les choses frustrantes (lenteurs de l'interface, etc)

Évolue en permanence au gré des discussions

Outils utilisés

Git et Gerrit

Mailing lists, forums, canaux IRC et XMPP

Trac

Github

Pootle et Haiku Userguide Translator

Doxygen

Autour de Haiku

Développer un système d’exploitation, c’est bien, mais…

- Applications (nouvelles ou récupérées de BeOS)
- Support/entraide
- Portage d’applications depuis Linux
- Contributions de patches
- Tests et rapports de bugs

Haiku, inc

Association basée aux USA

Habilitée à recevoir des dons

N’intervient pas dans les décisions techniques

Financement

Budget : 10000 à 30000€ par an

Principalement par des dons

- Utilisateurs
- Sponsoring de Google

Vente de DVDs

Dépenses

Serveurs, noms de domaine, infrastructure

Site web, dépôt git, distribution des mises à jour, buildbots, ...

Frais de déplacement pour des conférences, etc

« Coding sprints »

Contrats pour des développeurs freelance

Technique

Bonnes pratiques

Respect des règles de formatage

Privilégier d'abord la lisibilité, la performance ensuite

Corriger les problèmes plutôt que les contourner

Documentation

- Guide de l'utilisateur
- Référence pour les développeurs d'application
- Documentation interne du système

Le code est toujours maintenable au bout de 20 ans !

Choix techniques

Écrit en C++

Bon compromis entre lisibilité, performance, rapidité de développement

Utilisable dans toutes les couches (oui, même dans le noyau!)

Tous les composants sont développés par la même équipe

Permet de corriger les problèmes au bon endroit

Utilisation généralisée du multi-thread

Combien de threads pour un « Hello world » graphique ?

API et « Kits »

Objectif : fournir une API « clé en main »

Découpage en « kits »

Network Kit, Interface Kit, Storage Kit, Media Kit, …

Pas besoin d’autres dépendances

API de base commune

Possibilité d’extension

Translation Kit

Traiter les problèmes à la source

Pour bien faire marcher cette API, on a besoin de régler des détails à tous les niveaux

Un exemple ?

Exemple : création de threads

On doit donner une priorité

thread_id spawn_thread(thread_func func, const char* name, int32 priority, void* data);

Les priorités ont une sémantique

B_LOW_PRIORITY 5 ← Un compilateur

B_NORMAL_PRIORITY 10 ← Une application « classique »

B_DISPLAY_PRIORITY 15 ← Une fenêtre

B_URGENT_DISPLAY_PRIORITY 20

B_REAL_TIME_DISPLAY_PRIORITY 100 ← Un lecteur de vidéo

B_URGENT_PRIORITY 110

B_REAL_TIME_PRIORITY 120 ← Un lecteur de musique

Sous Linux :

On peut donner une priorité

pthread_create() + sched_setattr()

Les valeurs n'ont pas de sémantique

Est-ce que la priorité 10 d'une application est plus importante que la priorité 9 d'une autre ?

L'ordonnanceur ne peut qu'essayer de deviner...

Combiner les applications

Dans un shell UNIX, on a les pipes et c'est cool !

Mais on a pas la même chose pour les applications graphiques ?

Sous UNIX, tout est fichier

Mais le format du contenu n'est pas défini

BMessage

Un BMessage contient un… message !

« what » : identifiant sur 32bits

Données typées et nommées

Structuré par imbrication de messages

BLooper / BHandler

BLooper : une boucle de traitement d'évènements

Distribue les évènements aux BHandlers associés

BHandler : traite les messages

void MessageReceived(BMessage*);

Utilisé dans une application

Clic sur un bouton = message

Utilisé aussi entre applications

Copier-coller, drag and drop, ...

Sérialisation

On peut stocker un BMessage dans un fichier

Paramètres d'une application dans un fichier de configuration

On peut envoyer un BMessage sur le réseau

Communication avec des applications distantes

On peut archiver un objet C++ dans un BMessage

Sauvegarde de l'état d'une application

Transport d'une session d'un ordinateur à un autre

Replicants

On peut stocker le contenu d'une fenêtre dans un BMessage, et l'envoyer à une autre application

- « widgets » sur le bureau
- Icônes dans la barre des tâches
- Plug-ins de toutes sortes

« Scripting »

Protocole standardisé pour demander à une application ce qu'elle sait faire

Interfaces standard

cliquer sur le bouton « X » de la fenêtre « Y »

Interfaces spécifiques

ouvrir un fichier dans un éditeur de texte à la ligne 50, colonne 38

Fichiers et attributs

Attributs classiques

Nom, date de création, droits

Attributs étendus

Nommés et typés

Personnalisables selon le type du fichier

Indexation

Utiliser le système de fichiers comme une base de données

Mise à jour des résultats en temps réel

Exemples d’utilisation

E-Mails

Expéditeur, destinataire, sujet, statut, …

Le contenu du fichier proprement dit est juste le contenu du message

Tous les clients mail pour Haiku utilisent le même format

Musique

Artiste, album, genre, évaluation, ...

Une requête peut générer une playlist

Contacts, images, …

Icônes des fichiers (dans un format vectoriel compact)

Rejoignez-nous !

Pourquoi contribuer à un projet libre ?

Parce que vous l'utilisez

Parce que c'est une expérience professionnelle

Pour suivre un projet sur le long terme

Pour rencontrer des gens d'autres horizons