# Présentation Langage Red

Cht'i RUG 2011

# Les limites de REBOL

- Propriétaire, sources fermés
- Lent comparé aux autres langages (benchmark)
- Pas de support du multithreading
- Langage "glue", pas assez généraliste
- Difficilement intégrable dans des produits tiers
- Ne supporte pas les VM populaires (JVM, CLR)

# Introduction au langage Red

- Dialecte REBOL Red[uit]
- Open Source (BSD)
- Compilé statiquement + compilateur JIT
- Support de la programmation parallèle
- Généraliste (support de la programmation système)
- Utilisation en scripting comme REBOL (console)
- Aisément intégrable dans des applications tierces (Lua)
- Micro serveur web performant embarqué
- Développment en cours, commencé il y a 6 mois, public depuis 3 mois

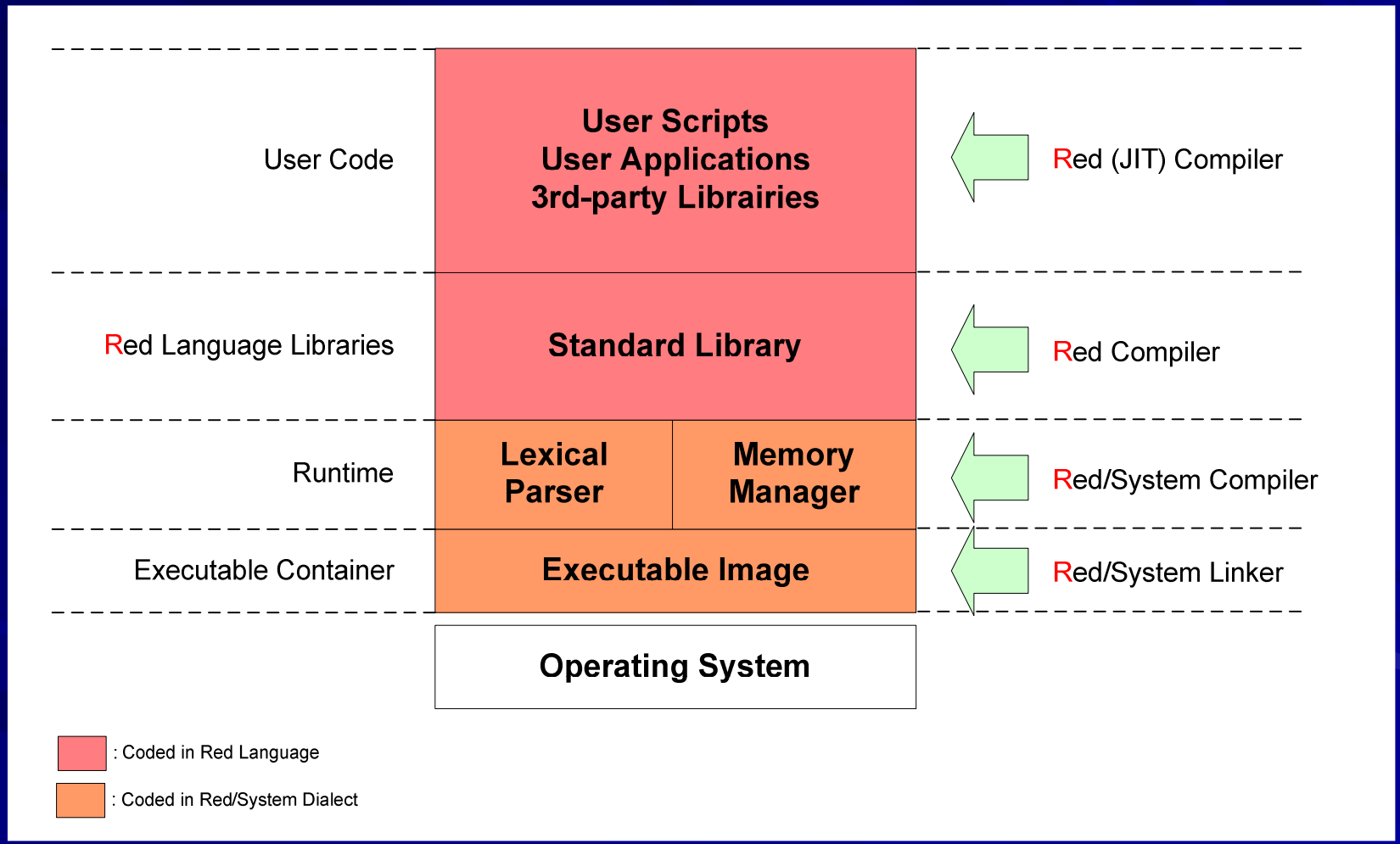
# Langage Red : Caractéristiques principales

- Syntaxe : fortement inspirée par REBOL
- Règles sémantiques : similaire à REBOL
- Système de type
  - riche, plupart des types REBOL, nouveautés (IPv6)
  - extensible par plugin (forme literale accessible ?)
  - inférence de type simple
  - erreurs de types détectés à la compilation plutôt qu'à l'exécution
- Fonctionnel impur mais avec support des HOF
- Support de la meta-programmation (compilation JIT)

# Aspects REBOL non supportés par Red

- Code trop "abstrait"
  - En règle générale, tout ce qui n'est pas statiquement compilable
  - foo: func [ a ][ a/b/c ]  => "a" peut être un object!, function!, block!,…
- "Binding" dynamique des mots
  - REBOL: peut changer la portée d'un mot dynamiquement (BIND)
  - Red v1.0: portée des mots (word!) statique uniquement
  - Règles de "Binding" dynamique (comme REBOL) possibles dans une couche d'émulation de plus haut niveau

# Vue globale de l'architecture de Red



# Modèle mémoire de Red

- Allocation mémoire "thread-local"
  - Tableau de cellules de 128-bit
- Possibilité d'avoir des structures de données partagées immuable (lecture seule)
- Garbage collector
  - Collecteur de type "compacteur"
  - Modèle du GC pour la v1.0: "stop-the-thread"
  - GC incrémental pour la v2

# Dialecte Red/System

- Purement imperatif, niveau langage C, syntaxe Red
- Compilé statiquement (compilation naïve en v1)
- Système de type limité:
  - 6 types: byte!, logic!, integer!, struct!, pointer!, c-string!
  - inférence de type limitée
- Assembleur "inline" et accès intégré aux ports E/S
- Linker
  - Types d'image finale: Exe, DLL, Lib
  - Formats: PE,  ELF, mach-o
- Cibles: IA-32, ARM, x64, JVM, CLR
- Red/System en tant que dialecte intégré de Red

# Programmation concurrente et // en Red

- "PPP challenge" (Intel)
  - Nous sommes maintenant dans l'ère des CPU multi-coeurs
  - Opportunité pour de nouvelles solutions / langages
- Parallélisation des tâches
  - Exécuter plusieurs morceaux de codes en // sur plusieurs coeurs
  - Red y répondra par une couche d'abstraction de type "Acteur"
- Parallélisation des traitements de données
  - Traiter une structure de donnée en utilisant plusieurs cœurs en même temps
  - Red y répondra en fournissant des collections parallèles

# Bootstrapping de Red (la poule et l'oeuf)

1) Ecrire le compilateur Red/System en REBOL
2) Ecrire le linker Red en REBOL
3) Ecrire le runtime Red en Red/System
4) Ecrire le compilateur Red en REBOL
5) Ecrire la librairie standard Red en Red
6) Réécrire le compilateur Red/System en Red
7) Réécrire le compilateur Red en Red
8) Ecrire le compilateur JIT de Red en Red
9) Si encore vivant, prendre des vacances! ☺

# IDE Red

- Indispensable pour beaucoup de développeurs
- Edition de code : composant Scintilla
- Focalisation sur les capacités de débogage
  - exécution pas-à-pas du code Red
  - exécution pas-à-pas des règles de Parse
  - capture des flux d'E/S pour inspection
- Profiler de code
- GUI en Red utilisant une abstraction type SWT
- Support des "bulles" de code (v2)

# Facteurs clés de succès de Red

- "Time to market"
  - Aussi court que possible
  - Itérations courtes (pas d'effet "tunnel" pendant des mois)
  - Indispensable pour réussir
- Communauté: atteindre une masse critique
  - Tenir la communauté informée (sites web, blog, twitter,...)
  - Faciliter les contributions (github)
  - Etre ouvert (éviter le syndrome de la tour d'ivoire)
  - Objectif: atteindre la masse critique (suffisamment de contributeurs)
  - Créer le buzz par une killer-app ou killer-demo
  - Occuper une ou plusieurs niches

# Avancement depuis 3 mois (1/2)

- Developpement
  - Red/System finalisé à 98%, ~2600 tests unitaires
  - Spécifications rédigées à 95% du langage Red/System
  - Base de code: 60Ko => 110Ko (~3200 lignes)
  - 164 commits sur Github dont 44 par les contributeurs
  - ~3500 pages vues sur le dépôt Github
- Nouveautés
  - Ajout des types byte!, logic! et pointer!
  - Inference de type sur les variables locales et la valeur de retour
  - Portage vers Linux
  - Portage Mac OSX en cours

# Avancement depuis 3 mois (2/2)

- Communauté
  - Support de la librairie 0MQ (Kaj de Vos)
  - Framework de tests dédié Quick-Test (Peter WA Wood)
  - 3 contributeurs sur github, 19 followers
  - Bugtracker: 61 tickets dont 53 traités
  - AltME: ~1500 messages dans le groupe Red
  - Blog: 8 articles, ~7000 pages vues
  - #red_lang : 124 tweets, 38 followers
  - Liste de diffusion: 31 inscrits, 44 sujets, 145 messages
  - IRC: 5 connectés en permanence

# Planning

- Sept. 2011:
  - beta de Red (sans JIT)
  - alpha du portage ARM
  - alpha de l'IDE
- Dec. 2011:
  - v1.0 de Red (sans JIT)
  - beta de l'IDE
- T1 2012:
  - beta du compilateur JIT pour Red
  - v1.0 de l'IDE

# Suivez moi sur…

- Blog de Red : http://red-lang.org
- Canal twitter de Red : #red_lang

…à très bientôt !