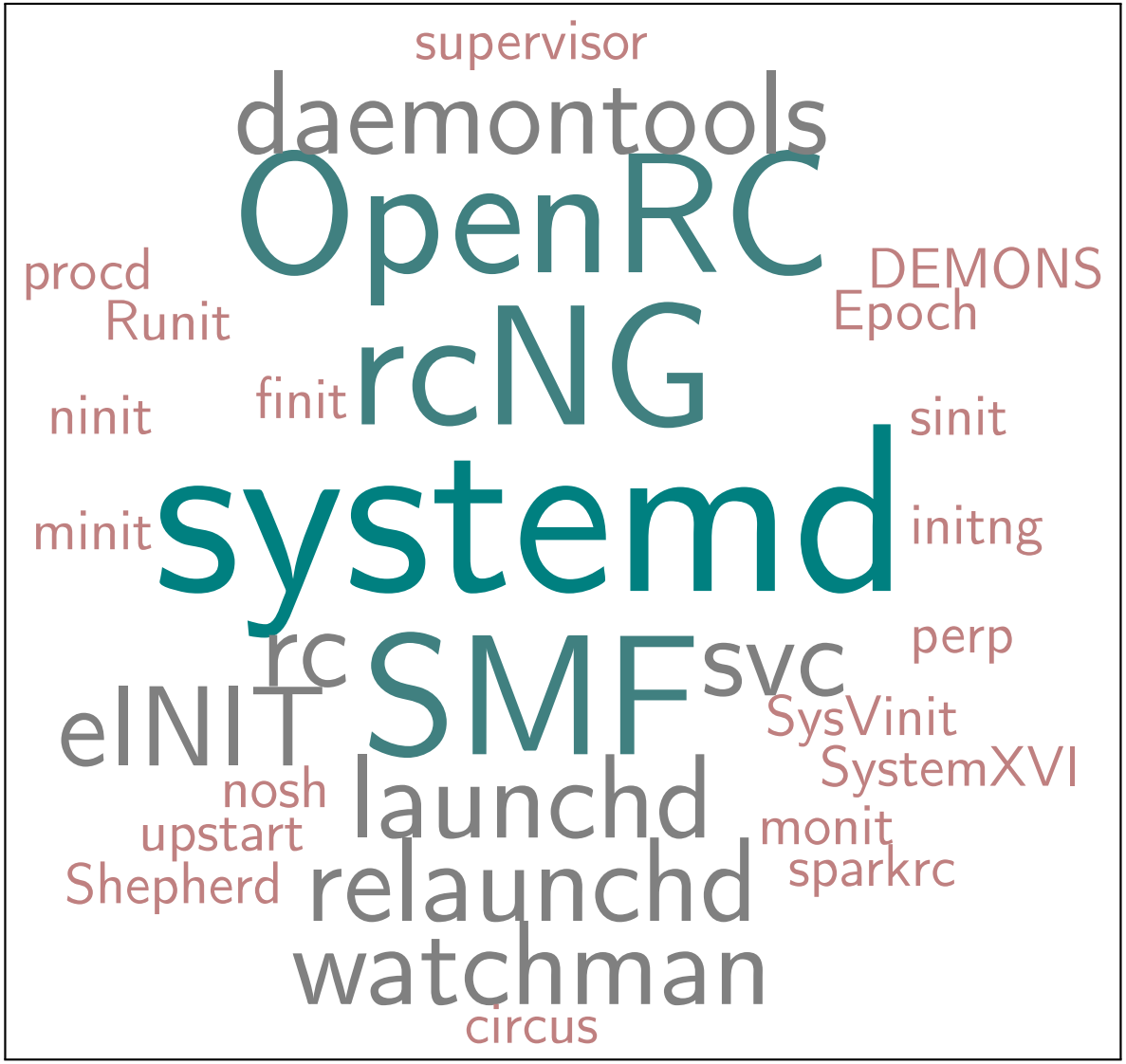
Gestion de services sous Unix

Baptiste Daroussin
bapt@FreeBSD.org
bapt@gandi.net

sysadmin #6
Paris
19 Février 2016

Disclaimer

L'arlésienne



gandi.net

Il était une fois init(8)

- petit mais costaud

Il était une fois init(8)

- petit mais costaud
- I'm your father(c)(tm)

Il était une fois init(8)

- petit mais costaud
- I'm your father(c)(tm)
- laisse la main a d'autres pour la gestion des services...

Il était une fois init(8)

- petit mais costaud
- I'm your father(c)(tm)
- laisse la main a d'autres pour la gestion des services... presque

Il était une fois init(8)

- petit mais costaud
- I'm your father(c)(tm)
- laisse la main a d'autres pour la gestion des services... presque
  - /etc/inittab
  - /etc/ttys

gandi.net

Il était une fois init(8)

- petit mais costaud
- I'm your father(c)(tm)
- laisse la main a d'autres pour la gestion des services... presque
  - /etc/inittab
  - /etc/ttys
- Généralement des scripts shell
  - SysVinit
  - rcNG
  - rc

Alors pourquoi tout ce bordel?

Alors pourquoi tout ce bordel?

- trop naïf

Alors pourquoi tout ce bordel?

- trop naïf
- framework complexes

Alors pourquoi tout ce bordel?

- trop naïf
- framework complexes
- cohérence compliquée

Alors pourquoi tout ce bordel?

- trop naïf
- framework complexes
- cohérence compliquée
- gestion de ressource complexe

Alors pourquoi tout ce bordel?

- trop naïf
- framework complexes
- cohérence compliquée
- gestion de ressource complexe
- peu réactif

Trop naïf

Trop naïf

- ne peut superviser qu'un process simple (via son pid)

Trop naïf

- ne peut superviser qu'un process simple (via son pid)
- perdu si double fork

Trop naïf

- ne peut superviser qu'un process simple (via son pid)
- perdu si double fork
- interactions limitées

Trop naïf

- ne peut superviser qu'un process simple (via son pid)
- perdu si double fork
- interactions limitées
- gestion de dépendance faible

Framework complexes

Framework complexes

- généralement en shell

Framework complexes

- généralement en shell
- complexe à isoler (cf commande service(8))

Framework complexes

- généralement en shell
- complexe à isoler (cf commande service(8))
- complexe à garder une cohérence entre les scripts d'init

Framework complexes

- généralement en shell
- complexe à isoler (cf commande service(8))
- complexe à garder une cohérence entre les scripts d'init
- complexe d'intégrer les systèmes de gestion de ressources

Framework complexes

- généralement en shell
- complexe à isoler (cf commande service(8))
- complexe à garder une cohérence entre les scripts d'init
- complexe d'intégrer les systèmes de gestion de ressources
- souvent inefficace (performances)

Framework complexes

- généralement en shell
- complexe à isoler (cf commande service(8))
- complexe à garder une cohérence entre les scripts d'init
- complexe d'intégrer les systèmes de gestion de ressources
- souvent inefficace (performances)
- pas de mécanismes de surveillance de processus

Framework complexes

- généralement en shell
- complexe à isoler (cf commande service(8))
- complexe à garder une cohérence entre les scripts d'init
- complexe d'intégrer les systèmes de gestion de ressources
- souvent inefficace (performances)
- pas de mécanismes de surveillance de processus
- peu de contrôle sur les processus (double fork)

gandi.net

Framework complexes

- généralement en shell
- complexe à isoler (cf commande service(8))
- complexe à garder une cohérence entre les scripts d'init
- complexe d'intégrer les systèmes de gestion de ressources
- souvent inefficace (performances)
- pas de mécanismes de surveillance de processus
- peu de contrôle sur les processus (double fork)
- fiabilité aléatoire du statut

Réactivité

Réactivité

- gestion événementielle forcément externe (cron, udev/devd, inetd)

Réactivité

- gestion événementielle forcément externe (cron, udev/devd, inetd)
  - duplication de la gestion des services
  - incohérence dans la gestion des services

Réactivité

- gestion événementielle forcément externe (cron, udev/devd, inetd)
  - duplication de la gestion des services
  - incohérence dans la gestion des services
- gestion événementielle souvent peu réactive: latence entre l'arrivée de l'évènement et l'action

Réactivité

- gestion événementielle forcément externe (cron, udev/devd, inetd)
  - duplication de la gestion des services
  - incohérence dans la gestion des services
- gestion événementielle souvent peu réactive: latence entre l'arrivée de l'évènement et l'action
- fault management absent

D'autre soucis: pas simple d'être un bon daemoniste

D'autre soucis: pas simple d'être un bon daemoniste

- utiliser setuid(2), setgid(2) ET setgroup(2) (Vérifier que l'opération c'est réellement effectuée)

D'autre soucis: pas simple d'être un bon daemoniste

- utiliser setuid(2), setgid(2) ET setgroup(2) (Vérifier que l'opération c'est réellement effectuée)
- toujours faire un chdir(2) après un chroot(2) (Vérifier leur bon fonctionnement)

D'autre soucis: pas simple d'être un bon daemoniste

- utiliser setuid(2), setgid(2) ET setgroup(2) (Vérifier que l'opération c'est réellement effectuée)
- toujours faire un chdir(2) après un chroot(2) (Vérifier leur bon fonctionnement)
- configurer de manière portable les setrlimit(2)

D'autre soucis: pas simple d'être un bon daemoniste

- utiliser setuid(2), setgid(2) ET setgroup(2) (Vérifier que l'opération c'est réellement effectuée)
- toujours faire un chdir(2) après un chroot(2) (Vérifier leur bon fonctionnement)
- configurer de manière portable les setrlimit(2)
- utiliser les mécanismes OS spécifiques de sandboxing

gandi.net

Le problème des PIDs

Le problème des PIDs

- concept "racy"

Le problème des PIDs

- concept "racy"
- services avec multiple workers hors de contrôle :
  - les fils qui échappent à la surveillance des parents
  - les parents laxistes qui abandonnent leurs enfants

Ça manque de sucre un peu tout ça

- y a des trous dans ma syslog :(
- c'est pas très parallèle tout ça
- comment je m'assure que ça marche?

Des solutions possibles?

- Pour les PIDs :

Des solutions possibles?

- Pour les PIDs :
  - process descriptors (pas suffisant) (Linux 4.1, FreeBSD 10.0)

Des solutions possibles?

- Pour les PIDs :
  - process descriptors (pas suffisant) (Linux 4.1, FreeBSD 10.0)
  - les reapers (Linux 3.4, FreeBSD 10.2, Dragonfly 4.0, Solaris 10)

Des solutions possibles?

- Pour les PIDs :
  - process descriptors (pas suffisant) (Linux 4.1, FreeBSD 10.0)
  - les reapers (Linux 3.4, FreeBSD 10.2, Dragonfly 4.0, Solaris 10)
- Gérer les ressources le sandboxing et la daemonisation via de génération de services

Des solutions possibles?

- Pour les PIDs :
  - process descriptors (pas suffisant) (Linux 4.1, FreeBSD 10.0)
  - les reapers (Linux 3.4, FreeBSD 10.2, Dragonfly 4.0, Solaris 10)
- Gérer les ressources le sandboxing et la daemonisation via de génération de services
- La réactivité : faire communiquer nativement les différents composants

Des solutions possibles?

- Pour les PIDs :
  - process descriptors (pas suffisant) (Linux 4.1, FreeBSD 10.0)
  - les reapers (Linux 3.4, FreeBSD 10.2, Dragonfly 4.0, Solaris 10)
- Gérer les ressources le sandboxing et la daemonisation via de génération de services
- La réactivité : faire communiquer nativement les différents composants
- La complexité : décrire au maximum les services via des format de description

Et un peu de sucre en poudre

Et un peu de sucre en poudre

- Préparer les sockets :
  - aide à la parallélisation
  - simplifie la séparation de privilèges
  - aide à la robustesse

Et un peu de sucre en poudre

- Préparer les sockets :
  - aide à la parallélisation
  - simplifie la séparation de privilèges
  - aide à la robustesse
- utiliser un IPC:
  - limite la duplication de code
  - permet la séparation des tâches (KISS)

Oui mais...

Oui mais...

- Faut il remplacer init(8) ?

Oui mais...

- Faut il remplacer init(8) ? pas nécessairement

Oui mais...

- Faut il remplacer init(8) ? pas nécessairement
- Un des systèmes existant permet il corrige t il tout ça ?

Oui mais...

- Faut il remplacer init(8) ? pas nécessairement
- Un des systèmes existant permet il corrige t il tout ça ? NON

Oui mais...

- Faut il remplacer init(8) ? pas nécessairement
- Un des systèmes existant permet il corrige t il tout ça ? NON
- Un des systèmes adresse t il les problèmes majeures ?

Oui mais...

- Faut il remplacer init(8) ? pas nécessairement
- Un des systèmes existant permet il corrige t il tout ça ? NON
- Un des systèmes adresse t il les problèmes majeures ? OUI

Oui mais...

- Faut il remplacer init(8) ? pas nécessairement
- Un des systèmes existant permet il corrige t il tout ça ? NON
- Un des systèmes adresse t il les problèmes majeures ? OUI
- Allez lequel ?

Oui mais...

- Faut il remplacer init(8) ? pas nécessairement
- Un des systèmes existant permet il corrige t il tout ça ? NON
- Un des systèmes adresse t il les problèmes majeures ? OUI
- Allez lequel ?

SMF

Oui mais...

- Faut il remplacer init(8) ? pas nécessairement
- Un des systèmes existant permet il corrige t il tout ça ? NON
- Un des systèmes adresse t il les problèmes majeures ? OUI
- Allez lequel ?

SMF

mais over engineer :(

Un peu de lecture

- Solaris Service Management Facility: Modern System Startup and Administration
- The Design and Implementation of the NetBSD rc.d system
- Les commentaires de Paul Vixie dans do_command.c

Questions?

gandi.net