Linux Embarqué

Réussir son projet

Gilles BLANC

RTS 2011, 31/03/2011

http://gillesblanc.com
contact@gillesblanc.com, gblanc@linagora.com

Licence

- **Creative-Commons by-nc**
- Couvre l'intégralité des présentes planches de présentation
- Paternité/pas d'utilisation commerciale sans autorisation
- <http://creativecommons.org/licenses/by-nc/2.0/fr/legalcode>

Au sommaire

- Éviter les erreurs fatales
- Évaluer le projet
- Mettre correctement en place le projet
- Mener à bien le projet

Éviter les erreurs fatales

Coûts cachés

- Première erreur : « le libre est gratuit »
- NON : certes pas de coûts *a priori*
- MAIS : tous les coûts sont cachés
  - Développement
  - Acquisition d'une solution sur étagère
  - Assistance technique
  - Surcoût lié à la complexité de la solution

Estimation des travaux

- Chausses-trappes
- Attention aux modules !
  - Complexité <=> Temps de développement
  - Disponibilité de la documentation, licences et NDA
  - Problème du propriétaire et « semi-ouvert »
- Attention aux temps de réponse !
  - Dépend du matériel
  - Dépend de la solution logicielle Linux
- Attention aux dépendances !
  - Système homogène et compilation
  - Packaging et patching

Disponibilité de compétences

- Linux nécessite plus de connaissances
- Connaissances transverses
- Ressources difficiles à trouver
  - 3 à 6 mois pour un dev'
  - Formation DIF longue à obtenir
  - Formation insuffisante, besoin d'expérience
  - Ressources geeks & pénibles à gérer
  - La blague de l'off-shore
- Prévoir un temps de lancement !

Licences

- Très spécifique, technique-juridique
- Une erreur est fatale
- Les ingénieurs s'en fichent
- Les chefs de projets ne sont pas au courant
- Les juristes sont débordés ou inexistants
- Sujet complexe : GPL, LGPL, Apache, noyau et modules Linux, bibliothèques, programmes propriétaires, etc.

Évaluer le projet

Périmètre

- Qu'est-ce qui existe déjà ?
  - Out of the box
  - À réadapter
- Qu'est-ce qui peut être intégré ?
  - Choix d'un framework ? (graphique, OE, buildroot, etc.)
  - Quelles implications sur l'architecture ?
- Qu'est-ce qui reste à développer ?
  - Sur le framework ?
  - Sur quoi s'appuyer ?

POC

- Mise en place de proof-of-concept
- Premier tour du périmètre
- Permet d'estimer la charge de travail
- Brouillon à réutiliser
  - Utiliser une solution Linux embarqué prête à l'emploi (Angström)
  - Coder en script (shell, Python)
  - Sans optimisation aucune
  - Sous émulateur ? (Qemu)

Estimation du temps

- En avant-vente puis au fur et à mesure de la réalisation
- « combien de temps ça va te prendre ? »
  - Définir le périmètre & specs ? 5 à 10 jours
  - Compiler un système Linux ? 10 à 15 jours
  - Intégrer un système Linux ? 5 jours
  - Optimiser ? 5 jours
  - Coder un module ? Heu... 10 à 40 jours

  (estimations non contractuelles à la louche)

Calcul des coûts

- Un ingénieur (junior) : ~750€/jour en SSII
  - Attention : forfait vs assistance technique & abus courants
- Un expert :
  - 1000 à 1500€/jour en SSII (dépend du nombre de jour et du fractionnement)
  - Moins en indépendant (très peu !)
- Une solution sur étagère : 5 à 20.000€
- Autres (modules proprios, licences commerciales, etc.) : à marchander

Nouveaux paradigmes

- Android, MeeGo & co
- Compatibles... avec eux-mêmes
- Plus qu'une simple architecture
- Impose le langage, le look'n'feel, la philosophie, l'infrastructure
- Le plus loin : Android (à ce niveau ce n'est quasiment plus du Linux)
- Donc : nouvelles compétences particulières à acquérir

Mettre correctement en place le projet

Compétences et rôles

- Qui fait quoi dans l'équipe ?
  - Un chef de projet (maître d'œuvre)
  - Un dev' kernel (?)
  - Un concepteur et intégrateur de la solution
  - Un dev' couches logicielles
  - De préférence l'un des trois premiers est un expert
- Demander son DIF à temps !

Parallélisation des tâches

- Compilation / intégration : série
  - 1 personne
  - garder la logique du système/homogénéité
- Travaux noyau : parallèle (sur cible ou non ?)
  - 1 personne par module/tâche spécifique
- Intégration application graphique
  - Par une ou plusieurs personnes
- Couche logicielle maison
  - à voir (souvent le concepteur/intégrateur)

Environnement de dev'

- Machine puissante (compilation), avec beaucoup d'espace disque
- Connexion Internet/réseau
- Dépôt de source (Git vs SVN)
- Forge (RedMine)
- Carte(s) de dev' (& associé : SD/MMC, lecteurs, alim, câbles série, adaptateur USB-série)
- Cible finale
- JTAG ?

Mener à bien le projet

La vraie vie (par l'ingénieur)

- Légos : assembler des briques
- Superglue : lier les logiciels pour répondre à la problématique du projet
- Scotch : truc et astuce pour que ça tienne
- Incantations mystiques (secret défense)
- Toute inversion est fatale
  - Le projet est mal conçu
  - La méthode de La Rache® a été trop utilisée

Cycles de dev'

- Méthode Agile revisitée
  - « Release early, release often » (en commercialement correct)
  - Cycles de développement, intégration client
  - Gras sur maigre
- Initialisation du projet
  - POC
  - Carte de dev / émulation
- Compilation et intégration du système
- Intégration valeur ajoutée et finalisation

Relations client / fournisseur

- Telephone early, telephone often
- Une livraison n'est pas aisée
  - Problème de la nature des livrables
  - Problème d'intégration matérielle
  - Prévoir un système d'installation facilitée sur cible
  - Mieux vaut des démos en réunion
- Attention à ne pas dévier du projet initial
- Tout tracer, attention aux délais

Gestion proactive

- Réagir aux erreurs de conception
  - Les systèmes de fichiers
  - Le firmware et les mises à jour
  - Phase de spec importante, demander l'aide d'un expert !
- Sous Linux, on peut rattraper (normalement)
  - Le système est assez puissant et flexible
  - Mais ça prend du temps...
- Scotch ?...
  - En dernier recours
  - Maintenabilité menacée

Informations

- Sources
  - Web, IRC, Mailing lists
- Méthodes
  - Google, DIF, relations
- Bibliographie
  - *Linux Embarqué*, Pierre Ficheux
  - etc. (cf. bonus)
  - Mon mien (à paraître) (quand j'aurai trouvé un titre)

Conclusion

- Sujet souvent pris trop à la légère
- Trop tard lorsqu'on se rend compte de la difficulté
- Peu facilement dévier et couter très cher
- Important de tout calculer (notamment les besoins hardware)
- Linux embarqué est un investissement à long terme
- Trop de projet pas dans les clous (techniquement ou juridiquement)
- Nécessité d'en prendre conscience

Fin (& bonus ?)

Bibliographie

- *Linux embarqué*, Pierre Ficheux, éd. Eyrolles
- *Building Embedded Linux Systems*, Yaghmour, éd. O'Reilly
- *Embedded Linux Primer*, Hallinan, éd. Prentice Hall
- *Embedded Linux System Design and Development*, Raghavan/Lad/Neelakandan, éd. Auerbach Publishers
- *Programmation système en C sous Linux*, Christophe Blaess, éd. Eyrolles

Aspects juridiques

- CPI 113-9 : droits d'auteur pour logiciel
- CPI 131-1 : cession globale
- C Civ 1626-1640 : garantie d'éviction
- C Trav 8231-1 : délit de marchandage
- *Droit et expertise des contrats informatiques*, Hubert Bitan, éd. Lamy