NORMES ET STANDARDS

SCRIPTS D'EXPLOITATION

Copyright BV Associates 2013 IMEPSIA™ est une marque déposée par BV Associates

SOMMAIRE

Copyright BV Associates 2013 IMEPSIA™ est une marque déposée par BV Associates

1 Introduction

Ce livre blanc a pour objet de proposer des normes et des standard pour l'écriture des scripts d'exploitation des systèmes ouverts. Cette norme, initialement rédigée pour des environnements Linux, reste applicable à d'autres systèmes d'exploitation et tout langange de programmation structurée.

2.1 Objectif

Avoir un nombre restreint de scripts génériques dans le but de :

1. améliorer la qualité du service
2. améliorer la sécurité d'exploitation des services de production informatique
3. accélérer et sécuriser le processus de mise en production
4. mutualiser les outils et les compétences de production

2.2 Utilisation des normes et standards de ce document

Les normes et standards définis dans ce document peuvent s'appliquer à toute application en production ou pré-production ou ayant un contrat de service défini.
Les normes et standards définis dans ce document respectent les critères d'exploitabilité et définissent le cadre de la mise en œuvre des objectifs décrits par les contrats de service.
Tout ajout, retrait ou modification de normes doit s'inscrire de façon cohérente dans ce document.
Les personnels concernés par ce document doivent avoir les moyens logistiques d'exercer les actions qui leur sont assignés.

Copyright BV Associates 2013 IMEPSIA™ est une marque déposée par BV Associates

La bonne application des normes décrites dans ce dossier nécessite de disposer d'un support de la part de l'éditeur permettant de résoudre les problèmes logiciels et de disposer d'un support d'information et de formation à jour.

2.3 Cas de dérogation aux normes

Des éléments spécifiques peuvent amener à ne pas appliquer certaines normes. Les cas de dérogations doivent être exceptionnels, motivés et exhaustivement décrits, puis acceptés explicitement par toutes les parties concernées.

Copyright BV Associates 2013 IMEPSIA™ est une marque déposée par BV Associates

3 Mécanisme des scripts d'exploitation

3.1 Fiche signalétique

Objectif

Définir en une seule fiche tous les éléments destinés à permettre l'exécution d'un script d'exploitation. L'objectif sous-jacent est de pouvoir extraire automatiquement toutes les informations pour générer automatiquement chaque script d'exploitation et le plan associé.

Norme

La définition des travaux à effectuer doit être décrite dans une fiche normalisée. Cette fiche est remplie successivement par les études & développements (à partir du modèle conceptuel des traitements par exemple) puis complétée par les intégrateurs et enfin par les ingénieurs d'exploitation. Tout ou partie de cette fiche doit se retrouver dans le dossier d'exploitation livré au pilotage.

Standard

Intitulé	Description
Description	Zone de commentaires décrivant l'action de ce job
Application	Zone de commentaires donnant l'application concernée
Ref_E&D	Zone de commentaires donnant la référence développement
Version	Zone de commentaires donnant le numéro de version de la fiche
Date_création	Zone de commentaires donnant la date de création de la fiche
Remarques	Zone de commentaires généraux
Responsable_E&D	Zone de commentaires E&D
Responsable_Prod	Zone de commentaires Production
Nom_ordonnanceur	Environnement = « Nom_environnement » Session (phase ou séquence) = « Nom_session »

Copyright BV Associates 2013 IMEPSIA™ est une marque déposée par BV Associates

	Job = « nom_job »
Nom_machine	Machine = «hostname | adresse IP»
Nom_OS	Script= «Nom_du_script_shell_associé [param]»
User_exécution	User= « nom_user_défini_dans_l'os »
Service Standard de Production	SSP_ROOT= « racine du SSP »
Pré-condition	Contraintes_horaires = « Nom_contrainte » Dépendances_prédécesseur = « job » « OK | Erreur » Dépendances_ressource = « nom_fichier » « present | absent » Dépendances_exclusion = « nom_job »
Action avant	Liste_Actions = "Nom_du_script [<paramètres>]"
Action	Liste_Actions = "Nom_du_script [<paramètres>]"
Action après	Liste_Actions = "Nom_du_script [<paramètres>]"
Action Si erreur	Si warning alors Liste_Actions = "Nom_du_script [<paramètres>]" Fsi Si erreur alors Liste_Actions = "Nom_du_script [<paramètres>]" Fsi Si erreur catastrophique alors Liste_Actions = "Nom_du_script [<paramètres>]" Fsi
Message Supervision	***<Date> <Hostname> %<Module>: <Source>:<Subsource>:<Index>:[<arg1>#<argN>]-<Severité>-<Identifiant>:<Valeur>:"<Texte>"***
Consigne si erreur	Durée : permanente | temporaire jusqu'à < condition > Consigne="Si conditionX alors action Y sinon action Z"

Copyright BV Associates 2013 IMEPSIA™ est une marque déposée par BV Associates

Condition d'escalade	Si « condition X » ou si « la consigne n'aboutit pas à la solution du problème » alors avertir X ou ne rien faire jusqu'à Xh puis avertir Y Fsi
Fichiers/tables manipulés (type = log | édition | data…)	Liste_Entrée= « (<nom_fichier_unix,type) » Liste_Sortie= « (<nom_fichier_unix,type) » Temporaire = « (<nom_fichier_unix,type) »
Flux	Lsite_Entrée = « (<nom_fichier_unix,type,id_transfert) » Liste_Sortie = « (<nom_fichier_unix,type, id_transfert) »
Temps moyen d'exécution	Zone de commentaires donnant le temps moyen d'exécution

Intitulé des contraintes horaires	Description
Nom_contrainte	Nom= « nom_de_la_contrainte »
Fréquence	Fréquence= « Journalier | Hebdomadaire | Mensuel | Annuel | Demande »
Horaire_déclenchement	Jour= «lundi | mardi | mercredi | jeudi | vendredi | samedi | dimanche » debut= «[0-4][0-9]h[0-9][0-9] » duree_planif=«[0-4][0-9]h[0-9][0-9] »
Règles_spécifiques	*Règle= «Zone de commentaires donnant des règles spécifiques de traitement -> impossible de le générer en automatique»*

Si la zone de « normes spécifiques » est remplie, la génération du plan ne pourra pas a priori être automatique.

Sinon, lorsque la fiche signalétique est remplie rigoureusement, on peut la définir selon une analyse lexicale et grammaticale et l'on est alors en mesure d'écrire un compilateur simple permettant de générer automatiquement les scripts.

Copyright BV Associates 2013 IMEPSIA™ est une marque déposée par BV Associates

3.2 Scripts d'exploitation

Définition fonctionnelle du script d'exploitation

Un script d'exploitation a pour objet d'assurer le lancement, l'exécution et la terminaison corrects d'une action clairement définie.

Définition intrinsèque du script d'exploitation

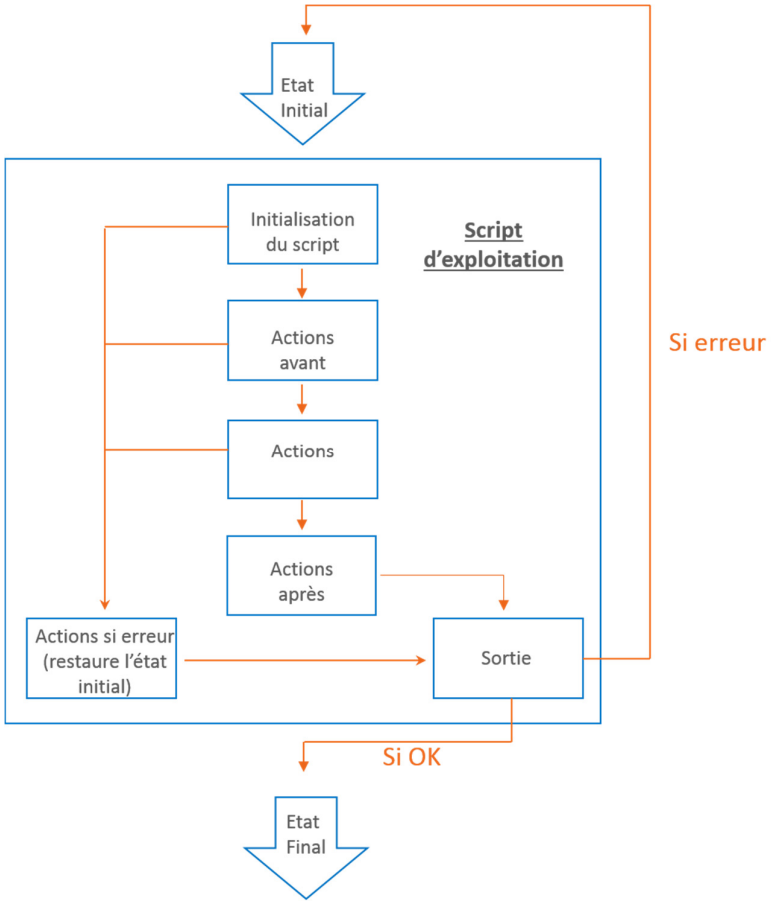
L'exécution d'un script d'exploitation est atomique : soit il exécute bien l'action d'exploitation et rend son environnement d'exécution dans l'état prévu, soit il rencontre un problème et rend son environnement dans l'état qui était celui de son lancement.
Un script d'exploitation garantit la traçabilité des opérations et des erreurs (dans un fichier de traces normalisé).
Un script d'exploitation garantit la possibilité de remonter des alarmes, en particulier par son code de terminaison (code retour cohérent avec les incidents éventuels rencontrés).

Normes de développement

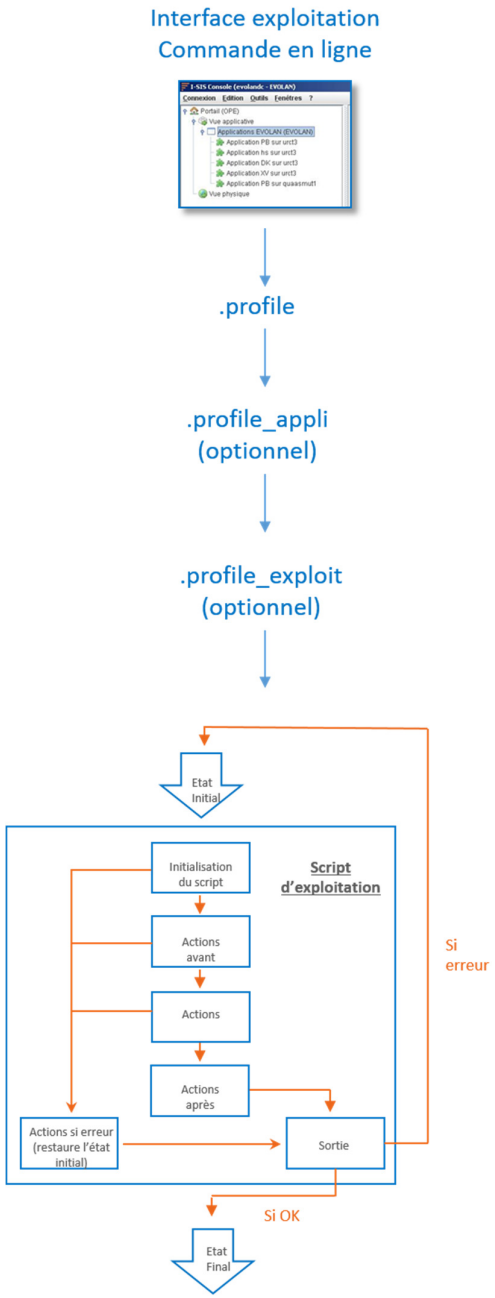
Tout script d'exploitation doit être accompagné d'une fiche signalétique décrivant exhaustivement ses caractéristiques d'exploitation et référencé dans un manuel technique.
Tout script doit avoir une programmation structurée permettant de clairement séparer code et données d'exploitation.
Tout script doit être aussi générique que possible (par exemple, un script général pour les arrêts/démarrage, statut, purge, backup, flux, des scripts de sauvegarde réutilisables selon les SGBD, …).
Tout script doit avoir un marquage de version analysable automatiquement.

Toutes les instanciations de variables extérieures au script doivent être faites lors de l'exécution d'un .profile général (exécuté à la connexion du user) appelant si nécessaire des profiles spécifiques.
Tout script doit donner un minimum d'informations d'usage.
Tout script d'exploitation devra respecter l'architecture et la dynamique suivante :

Copyright BV Associates 2013 IMEPSIA™ est une marque déposée par BV Associates


Etat
Initial
Initialisation
du script
Script
d'exploitation
Actions
avant
Actions
Actions
après
Actions si erreur
(restaure l'état
initial)
Sortie
Si erreur
Si OK
Etat
Final

Dynamique d'exécution d'un script



Objectif

Avoir un niveau de qualité de programmation en accord avec l'état de l'art

Normes

Tout script d'exploitation doit respecter un minimum de règles de programmation:
- spécifications préalables du script (N&S)
- versionning
- programmation procédurale et structurée
- séparation du code et des données
- structuration des données utilisées
- détection et gestion des erreurs
- traçabilité des actions (surtout en phase de mise au point)
- maîtrise du script vis à vis de son environnement (gestion des effets de bord)
- réutilisabilité du script dans des contextes différents (une fonction = une implémentation)
- documentation technique associée

Standard : outillage

La programmation doit être assistée par un outillage adéquat.

Traçabilité des changements

Objectif

Avoir des éléments de traçabilité des changements

Normes

Tout script doit est complété par des informations de maintenance.
Tout script doit être versionné.

Standard : Entête de fichier ksh

Copyright BV Associates 2013 IMEPSIA™ est une marque déposée par BV Associates

La partie ENTETE doit être renseignée de la façon suivante (marquage SCCS) :

```
#!/bin/ksh
##############################################################################
# @(#) $Internal Name: Nom du script$ - $Type: Korn shell script$
# @(#) $Revision: Revision$ - $Revision Date: Date$
# @(#) $Summary: Description rapide$
# @(#) $Author: Nom de l'auteur$
# @(#) $Fichiers in :liste$
# @(#) $Fichiers out : liste $
# @(#) $Fichiers temp : liste $
##############################################################################
# @(#) $Usage: Nom du script [arguments ...]
# Revision history:
# ----------------
# Revision      Who           When  Object
# --------              ---           ----    ------
#  1.0          Initiales             Date    Creation
#  Rev          Initiales             Date    Description
#-----------------------------------------------------------------------------
```

Cette entête est utilisable pour tout type de fichier Unix, compilé ou pas(C, ASCII, HTML, ...)

Contrôle du lancement

Objectif

Contrôler le lancement du script

Norme

Tout script d'exploitation doit contrôler que son appel est correct (options, paramètres).

Documentation du script

Objectif

Avoir un niveau minimum d'aide en ligne

Norme

Tout script d'exploitation doit permettre d'accéder à un usage indiquant sa syntaxe et une définition minimum de son action.

En cas d'erreur à l'appel du script, l'usage doit être rappelé.

Standard

L'usage sera appelé par l'option –h du script.

Fonctions standard d'un script d'exploitation ksh

Fonction standard de sortie : la fin d'un script doit obligatoirement passer par cette fonction de façon à retourner un code normalisé à l'environnement. On pourra introduire dans cette fonction la remontée systématique d'alerte à l'ordonnanceur. On loguera aussi systématiquement la fin d'exécution dans une logue centrale $LOG_ROOT/stdout.log selon le format :

« AAMMJJhhmmss | PPID : ? ? | PID : ? ? | nom_script | Type : Fin | Severite : ? ? | message d'information »

Fonction standard de vérification des codes retours : toute commande doit être testée, et toute erreur loguée. C'est le but de cette fonction.

Fonction standard d'action avant fonction principale : correspond au champ « action avant » de la fiche signalétique, elle ne contient que des actions d'exploitation (copie, backup, …).

Fonction standard d'action: correspond au champ « action» de la fiche signalétique, elle contient toutes les actions fonctionnelles (exécution des programmes fonctionnels).

Fonction standard d'action si erreur : correspond au champ « si erreur » de la fiche signalétique.

Fonction standard d'effacement des fichiers temporaires : tous les fichiers ou tables déclarés comme temporaires doivent être détruits dans cette fonction.

Copyright BV Associates 2013 IMEPSIA™ est une marque déposée par BV Associates

Fonction standard d'action après fonction principale : correspond au champ « action avant » de la fiche signalétique, elle ne contient que des actions d'exploitation (copie, effacement, …).

Fonction principale d'initialisation : un log de traitement doit être instancié, les variables indispensables à l'environnement doivent être instanciées, les profils d'exploitation spécifiques doivent être instanciés, une variable donnant l'usage du script doit être instanciée, les paramètres du script doivent être récupérés de façon standard. On loguera aussi systématiquement le début d'exécution dans une logue centrale $LOG_ROOT/stdout.log selon le format :

« AAMMJJhhmmss | PPID : ? ? | PID : ? ? | nom_script | Type : Debut | Severite : 0 | message d'information »

Exemple :

```
# !/bin/ksh
F_sortie()
{       (…)
}
F_analyse_code_ret()
{       (… )
}
F_rm_temporaire()
{       (… )
        si erreur alors warning
        fsi
}
F_si_erreur()
{       (… )
   F_rm_temporaire
   F_sortie
}
F_action_avant()
{       (… )
   si erreur alors
              F_si_erreur
         Fsi
}
F_action_apres()
{       (… )
        si erreur alors
              #on continue, pas de restauration
        Fsi
}
F_action()
```

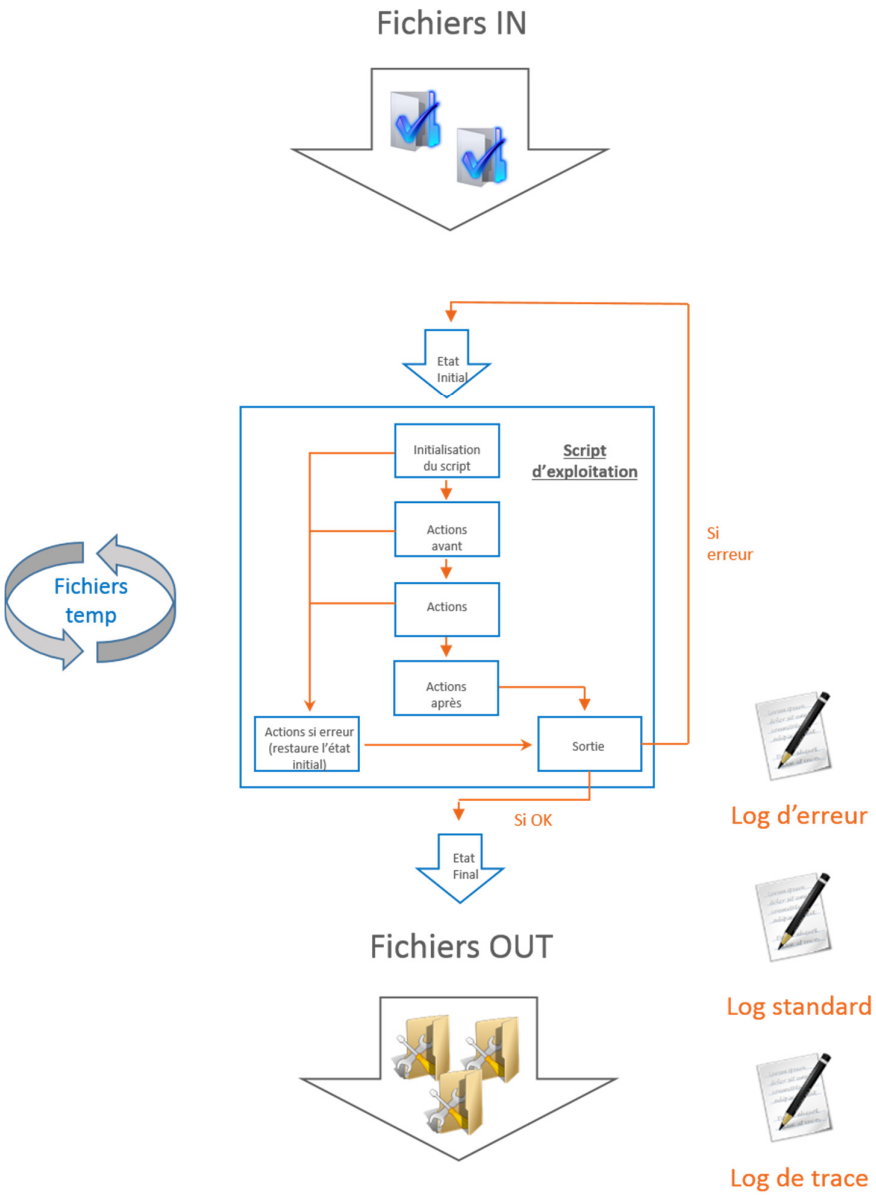
```
{       (… )
        si erreur alors
                F_si_erreur
        Fsi
}
F_init_env()
{       (… )
        si erreur alors
                F_sortie
        Fsi
}
#corps principal de la fonction
F_init_env
F_action_avant
F_action
F_action_apres
F_rm_temporaire
F_sortie
#fin du script
```

Standard : script nécessitant des paramètres

Certains scripts nécessitent le passage de paramètres. Par exemple :
< Mon_Script valeur_param1 valeur_param2 >

Nativement, certains logiciels passent des paramètres à un script à partir d'instanciation de variables d'environnement. On tient facilement compte de ceci avec une norme simple d'écriture des procédures d'exploitation (pour un langage de script de la famille des bourne shell).
L'usage de paramètres sur des scripts « universels » est important du point de vue de l'industrialisation car elle permet d'utiliser un unique script par grande fonction d'exploitation (statut, changement d'état, backup, livraison) et elle restreint le besoin de développement et de validation des scripts courants d'exploitation (purge, lancement de scripts sql,…).
#début du script
(…)
#Récupération des paramètres
PARAM1=${1 :-$PARAM1}
PARAM2=${2 :-$PARAM2}
(…)
#Fin du script

Standard : fichiers liés à l'exécution d'un script



3.3 Log normalisée de début de script

Objectif

Savoir gérer les messages de démarrage d'un script. Avoir une gestion technique normalisée de l'écriture du démarrage d'un script, automatisée et sécurisée.

Norme

Tout démarrage de script doit être logué par une méthode standard d'exploitation.

Standard

Méthode Log_Start.

Cf. « manuel de référence I-TOOLS ».

3.4 Log normalisée de fin de script

Objectif

Savoir gérer les messages de fin d'un script. Avoir une gestion technique normalisée de l'écriture de la fin d'un script, automatisée et sécurisée.

Norme

Tout arrêt de script doit être logué par une méthode standard d'exploitation.

Standard

Méthode Log_End

Cf. « manuel de référence I-TOOLS ».

Copyright BV Associates 2013 IMEPSIA™ est une marque déposée par BV Associates

3.5 Gestion des erreurs dans les scripts

Objectif

Savoir gérer les erreurs en cours de script.

Les scripts peuvent rencontrer un certain nombre de problèmes lors du traitement. Il est nécessaire d'avoir une gestion standardisée de cette gestion. Cette gestion peut en grande partie être reportée dans la fonction standard de vérification du code retour d'une action dans un script.

Norme

La gestion des erreurs au sein d'un script est de la compétence de l'ingénieur réalisant le script (test des commandes passées, définition du niveau de sévérité, …).

La remontée des erreurs vers le superviseur est de la compétence de l'ingénieur réalisant le script.

Le code retour de toute commande passée dans un script d'exploitation doit être testé.

Tout code non nul en retour d'une commande doit être logué par un script standard Log_Error
Tout message logué doit l'être au moins dans une log standard selon un format standard, éventuellement dans une log spécifique au script

Les différents codes d'erreur doivent être réduits à un nombre fini et standard de codes correspondant à des niveaux normalisés de gravité (i.e. warning = 201, erreur = 202 et erreur nécessitant l'arrêt de la production = 203)

Tous les messages significatifs (202, 203) doivent être regroupés dans une log écoutée par le superviseur éventuel.

Le code de terminaison du script doit être cohérent vis à vis des erreurs rencontrées.

Copyright BV Associates 2013 IMEPSIA™ est une marque déposée par BV Associates

3.6 Gestion des signaux d'abort dans les scripts

Objectif

Savoir gérer les signaux d'abort en cours d'exécution

Les scripts peuvent recevoir un signal d'abort (2 ou 15 sous UNIX et Linux) en cours d'exécution.

Norme

Les signaux d'abord doivent être « trappés » et l'exécution redirigée vers la fonction de traitement en erreur.

3.7 Log normalisée d'erreur

Objectif

Savoir gérer les messages d'erreurs générés par un script.

Avoir une gestion technique normalisée de l'écriture dans la log d'erreur, automatisée et sécurisée.

Norme

Toute erreur doit être loguées par une méthode standard d'exploitation.

Standard : méthode Log_Error

Cf. « manuel de référence I-TOOLS ».

3.8 Analyse normalisée des fichiers de trace

Objectif

Savoir gérer les erreurs générées dans un fichier de trace

Avoir une gestion technique automatisée, normalisée et sécurisée de l'analyse des erreurs contenues dans un fichier de log.

Norme

L'analyse des erreurs d'un fichier de trace doit être faite par une procédure normalisée.

Standard

Description : le programme analyse_log est un programme d'analyse standard de log s'appuyant sur deux tables (fichiers plats Unix) :

- err_significatives_NOM-APPLI
- err_non_significatives_NOM-APPLI

Chaque ligne de ces tables est un filtre qui donne le pattern (syntaxe bourne shell) de l'erreur à chercher où à ne pas retenir.

On peut écrire autant de tables de filtre que l'on veut. Les tables sont des données vivantes d'exploitation dont le contenu initial est défini par les études/développements : elles changent au gré des évolutions de la production mais leur utilisation au niveau de l'ordonnanceur est définie une fois pour toutes.

Le programme va d'abord extraire du fichier de log toutes les erreurs données comme non significatives et générer un fichier temporaire. Dans ce fichier temporaire, on cherchera toutes les données jugées significatives. S'il en existe, on les affiche et on renvoie un code d'erreur 202.

Ligne de commande :

Copyright BV Associates 2013 IMEPSIA™ est une marque déposée par BV Associates

analyse_log [-h] <nom_table_err_signicative> <nom_table_err_ nonsignicative> <nom_fichier_log > [param]

Paramètres :

< nom_table_err_signicative > indique qu'il faut aller chercher les patterns des erreurs significatives dans la table < nom_table_err_signicative >_rejet.

< nom_table_err_ nonsignicative> indique qu'il faut aller chercher les patterns des erreurs significatives dans la table < nom_table_err_nonsignicative >_accept.

Options :

-h donne la syntaxe de la commande

Affichage en sortie :

- Les éventuels messages d'erreurs

Code de sortie :

0 si OK

202 si erreur

Copyright BV Associates 2013 IMEPSIA™ est une marque déposée par BV Associates

4 Produire des scripts génériques

4.1 Gestion des données dans les scripts

Objectif

Savoir séparer le code et les données d'exploitation

Normes

Les données d'exploitation doivent être stockées de façon cohérente dans des tables identifiées.

Les données d'exploitation doivent pouvoir être facilement extraites de ces tables et transformées en variables utilisables par le langage natif du script.

Les données qui ne sont pas stockées dans les tables doivent être des données communes à l'exploitation et elles doivent être stockées dans des variables (internes au script ou partagées par différents environnements).

Standard

Méthodes Select et Read_Row

Cf. « manuel de référence I-TOOLS ».

4.2 Structuration des données

Objectif

Savoir décrire une structure de données

Copyright BV Associates 2013 IMEPSIA™ est une marque déposée par BV Associates

Normes

Les tables d'exploitation doivent être décrites structurellement pour permettre de les manipuler selon un format fixe et cohérent.
Ce modèle de données doit permettre de concevoir des scripts qui s'affranchissent d'implémentations spécifiques.

Standard : méthode Define_Table

Cf. « manuel de référence I-TOOLS »..

Copyright BV Associates 2013 IMEPSIA™ est une marque déposée par BV Associates