# Formation Debug Kernel

## Description de la formation Debug Kernel

Le point commun de toute plate-forme exploitée sous Linux, serveur, poste de travail, matériel embarqué, est sans aucun doute le noyau Linux. Les tests ou la mise en oeuvre de telles plate-formes dans des conditions plus ou moins extrêmes ou sur du matériel plus ou moins exotique conduisent assez fréquemment à des situations de blocage partiel (oops) ou total (panic) du noyau.

Cette **formation Debug Kernel** propose d'explorer avec le participant le système qui sous-tend le fonctionnement du noyau pour mieux l'appréhender et connaître les sources d'information liées.

Elle propose également des outils et des méthodes pour collecter les informations nécessaire à la dernière phase qui consiste en l'analyse du problème rencontré. Le participant est alors capable soit de corriger le dysfonctionnement soit de transmettre l'ensemble de ces informations au niveau compétent en faisant ainsi gagner du temps sur cette phase d'analyse.

## Objectifs

**Objectifs opérationnels :**

Collecter de manière exhaustive les informations liées à un dysfonctionnement du noyau et analyser les informations ainsi recueillies.

**Objectifs pédagogiques :**

Concrètement, cette **formation Debug Kernel** vous apporte les connaissances et compétences nécessaires pour :

- Connaître les sources d'information relatives au fonctionnement du noyau Linux
- Savoir collecter de manière exhaustive les informations liées à un dysfonctionnement du noyau
- Savoir analyser les informations ainsi recueillies

## À qui s'adresse cette formation ?

**Public :**

Ce cours Debug Kernel cible principalement les développeurs Linux.

**Pré requis :**

Pour suivre cette formation Debug Kernel dans des conditions optimales, il est nécessaire d'avoir une certaine connaissance du système Linux, ainsi que des connaissances de base en langage C.

## Contenu du cours Debug Kernel

### Systèmes de fichiers et debug

Système de fichiers virtuel procfs
Système de fichiers virtuel sysfs
Collecter des informations de debug avec debugfs
Stocker des informations de manière persistante avec pstore

### Debug user space

Récupérer un core dump
Utiliser gdb
Détection de head corruption avec heap / alloc

### Erreurs kernel et dialogue avec le noyau

cktrace

warn

Kernel tainted – liste des flags

oops

panic

bug

## Configurer son kernel pour améliorer le debug

debug info

kdump / kexec

Configuration de spin lock, mutex, utilisation de locks

printk

## Les outils de debug kernel

system.map

Mettre en place une console série

Spécificités de l'utilisation d'une console série sous Xen

Mise en place d'une netconsole

Utiliser qemu pour debugger

kgbd (port série)

crash / kdump

De l'importance de l'appareil photo

Tracing / ftrace

Quelques paramètres kernel utiles :

panic=oops, vga=, earlyprintk=, ignore_loglevel, initcall_debug, log_buf_len

## Analyser les informations recueillies

Identifier des adresses mémoire avec addr2line

gdb, le couteau suisse du débogage

Un outil d'analyse dédié au kernel : crash

Outil d'aide à l'analyse : printk

Définir un format de message avec pr_*

Extraire le device et son driver avec dev_*

printk versus dev_* ?