Android　Initブートシーケンス

カーネルが起動後に/initを実行する。ソースは(system/core/init/init.c)

No	概要	コマンド	関数	ファイル
1	変数初期化			init.c
2	initから起動した子プロセスが停止したり（子プロセスが SIGSTOP, SIGTSTP, SIGTTIN, SIGTTOU を受けたとき）再開したり（子プロセスが SIGCONT を受けたとき）したときに SIGCHLD の通知を受けないように設定する。	sigaction(SIGCHLD, &act, 0);		init.c
3	新規作成用ファイルのパーミッションを666に設定する。	umask(0);		init.c
4	/dev、/proc、/sysを作成。パーミッションは0755 /devにRAMDISK（メモリ上のファイルシステム）をマウントする /dev/pts、/dev/socketを作成。パーミッションは0755 /dev/ptsにRAMDISK（メモリ上のファイルシステム）をマウントする。 /proc、/sysにRAMDISK（メモリ上のファイルシステム）をマウントする。 上記はinitrdの処理。Initrd(initramdisk)は、実際のルート・ファイル・システムが使用できるようになる前にマウントされる初期ルート・ファイル・システム。柔軟にカーネルモジュール(*.ko)を組み込むための仕組みですが、ハードウエア構成が決まっていれば不要です。ソースのコメントによれば、kmsgとNULLのためにdeviceノードを作成しています。/devにtmpfsをマウントして外部とのやりとりが可能になります。 Solaさんのサイトを参考にしてビルドしたカーネルをlsmodで確認するとPowerVR用のモジュール（omaplfb.ko、pvrsrvkm.ko）のみ組み込んでいます。	mkdir mount mkdir mount mount		init.c
5	mknodeでスペシャルファイル”/dev/_null_”を作成する。作成後オープンしてFD（ファイルディスクリプタ）の0：標準入力、1：標準出力、2：標準エラー出力に割り当てる	mknod	open_devnull_stdio()	init.c
6	mkmodeでスペシャルファイル”/dev/_kmsg_”を作成する。作成後オープンして、FDにFD_CLOEXECを設定する。 オープン済みだったら、クローズするように設定	mknod	log_init()	util.c
7	設定ファイル：/init.rcを解析する。構造体：parse_stateに解析結果を設定する。アクションキューへのactionの登録をする。		parse_config_file(″/init.rc″)	parser.c
8	qemu_perms(構造体perms_の配列）を初期化する。		qemu_init()	devices.c
9	/proc/cmdlineの読込。″=″をセパレータとして値を取り出して内部変数にコピーする。取り出す値は以下の通り qemu androidboot.console androidboot.mode androidboot.serialno androidboot.baseband androidboot.carrier androidboot.bootloader androidboot.hardware 今回の場合は、以下の１行が”/proc/cmdline”に格納されている。 console=ttyS2,115200n8 noinitrd root=/dev/mmcblk0p2 rootfstype=ext3 rw rootdelay=1 init=/init		import_kernel_cmdline(0) import_kernel_nv	init.c

10	/proc/cpuinfoの読込。HardwareとRevisionから”: ”をセパレータとして値を取り出して内部変数にコピーする。 Hardwareはさらに空白が見つかるまで探して、小文字に変換する。(**omap3**が取得されるはず) /proc/cpuinfoの中身は以下の通り。 Processor : ARMv7 Processor rev 2 (v7l) BogoMIPS : 501.02 Features : swp half thumb fastmult vfp edsp neon vfpv3 CPU implementer : 0x41 CPU architecture: 7 CPU variant : 0x3 CPU part : 0xc08 CPU revision : 2 **Hardware** : OMAP3 Beagle Board **Revision** : 0020 Serial : 0000000000000000		get_hardware_name()	init.c
11	設定ファイル:init.Hardware.rcを解析する。HardWareは、get_hardware_name()で取得した名称BeagleBoardであれば、**init.omap3.rc**。構造体:parse_stateに解析結果を設定して、アクションキューへのactionの登録する。 内容は以下の通り on boot setprop ro.radio.use-ppp no setprop ro.radio.noril yes setprop net.eth0.dns1 8.8.8.8 setprop net.dns1 8.8.8.8 #LAN #service lan-setup /system/etc/init.omap3.sh #oneshot		parse_config_file(″init.omap3.rc″)	init.c
12	”early-init”という名前で関数:action_add_queue_tail()をキューに登録。登録されていなければ実行(今回は初回)する。		action_for_each_trigger action_add_queue_tail()	parser.c
13	アクションキューに登録されたコマンドをキューから取り出し、削除、実行。アクションキューが空になるまで実行する。init.rc, initomap3.rcに記述されたコマンドを実行します。		drain_action_queue()	init.c
14	デバイスの初期化:PF_NETLINKソケットを生成してbind実行。このソケットはカーネルからユーザプロセスへの通知(プロセス間通信)に使用する。 ソケットに対しては以下の設定を行う。 fcntl(fd, F_SETFD, FD_CLOEXEC); exec()を実行したらクローズする。(開けっ放しにしない) fcntl(fd, F_SETFL, O_NONBLOCK); 非ブロッキングモード設定。ソケットをRead(Recv)の際にデータがなければエラーを返すようにする。読み出せるまでサスペンドしないようにする。 /sys/class, /sys/block, /sys/device配下にueventファイルを見つけたら、ueventファイルに″add¥n″を書き込む。 その後、PF_NETLINKソケットに対してrecvを実行する。(非ブロッキングモードにしているので、データがなければエラーとなって処理を抜ける)。データを受信したら、メッセージをパースして、/sys、/dev配下にファイル作成		device_init()	devices.c
15	Anonymous Shared Memoryの設定。 スライド参照。		property_init()	property_service.c
16	input_keychord構造体の内容を/dev/keychordに書き込み		open_keychord()	init.c
17	変数:consoleからコンソール用のデバイスファイル名を決定。/dev/%s 決定したデバイスファイルをオープン	open()		init.c

18	/initlogo.rleを読み込んで起動ロゴを表示 ということは、起動ロゴを置き換えてやれば、好きな画像が出せるってことですね。 http://dev-dolphin1.seesaa.net/article/114264263.html に詳細があります。		load_565rle_image	logo.c
19	/dev/tty0をopenし、"ＡＮＤＲＯＩＤ"という文字列を書込んでclose。画面に青い文字でＡＮＤＲＯＩＤって出てた気がします。			init.c
20	QUMU（エミュレータ）で動作しているときは、頭に"ro.kernel."を付けてカーネルオプションをプロパティに設定する。 ro.factorytest：bootmodeがfactoryなら1、factory2なら2、それ以外なら0を設定 ro.serialno：変数serialnoが有効なら変数の値、無効なら""を設定 ro.bootmode：変数bootmodeが有効なら変数の値、無効なら"unknown"を設定 ro.baseband：変数basebandが有効なら変数の値、無効なら"unknown"を設定 ro.carrier：変数carierが有効なら変数の値、無効なら"unknown"を設定 ro.bootloade：変数bootloaderが有効なら変数の値、無効なら"unknown"を設定 ro.hardware：変数hardwareの値を設定 ro.revision：変数revisionの値を設定		import_kernel_cmdline(1) import_kernel_nv property_set	init.c
21	トリガー名："init"で関数：action_add_queue_tailを登録、実行。		action_for_each_trigger	init.c
22	アクションキューに登録されたコマンドをキューから取り出し、削除、実行。アクションキューが空になるまで実行する。		drain_action_queue()	init.c
23	PROP_PATH_SYSTEM_BUILD(/system/build.prop) PROP_PATH_SYSTEM_DEFAULT(/system/default.prop) PROP_PATH_LOCAL_OVERRIDE(/data/local.prop) を読み込んで、プロパティに設定する。		start_property_service()	property_service.c
24	双方向通信用のソケットを生成する。sigchld handlerが送受信すためのもの		socketpair	init.c
25	チェック処理：device_fd、property_set_fd、signal_recv_fdが正常に取得できていなければinit失敗で終了			init.c
26	全てのプロパティのトリガーをアクションキューに登録		queue_all_property_triggers	parser.c
27	アクションキューに登録されたコマンドをキューから取り出し、削除、実行。アクションキューが空になるまで実行する。		drain_action_queue()	init.c
28	device_fd、property_set_fd、signal_recv_fdに対してイベント：POLLIN（ブロックせずに読み出し可能）を設	poll		init.c
	device_fdにPOLLINがあった場合は、ソケットをRecv property_set_fdにPOLLINがあった場合は、ソケットをRecv signal_recv_fdはsignal_recv_fdをRead	recv Read		